

(19)



(11)

EP 3 851 956 A1

(12)

EUROPEAN PATENT APPLICATION
published in accordance with Art. 153(4) EPC

(43) Date of publication:

21.07.2021 Bulletin 2021/29

(51) Int Cl.:

G06F 9/445 (2018.01)

(21) Application number: **19872828.9**

(86) International application number:

PCT/CN2019/111015

(22) Date of filing: **14.10.2019**

(87) International publication number:

WO 2020/078314 (23.04.2020 Gazette 2020/17)

(84) Designated Contracting States:

AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

Designated Extension States:

BA ME

KH MA MD TN

(72) Inventors:

- **LI, Tao**
Shenzhen, Guangdong 518129 (CN)
- **YAO, Wanglai**
Shenzhen, Guangdong 518129 (CN)
- **YE, Fei**
Shenzhen, Guangdong 518129 (CN)

(30) Priority: **15.10.2018 CN 201811198385**

(74) Representative: **Thun, Clemens**

Mitscherlich PartmbB
Patent- und Rechtsanwälte
Sonnenstraße 33
80331 München (DE)

(71) Applicant: **HUAWEI TECHNOLOGIES CO., LTD.**
Shenzhen, Guangdong 518129 (CN)

(54) **METHOD AND APPARATUS FOR ACCELERATING COLD-STARTING OF APPLICATION, AND TERMINAL**

(57) This application records a method, an apparatus, and a terminal for accelerating cold startup of an application. The method for accelerating the cold startup of the application includes: after identifying an event that instructs an operating system of the terminal to cold start up an application, obtaining, by the terminal from a plurality of dimensions, current status information related to the cold startup of the application, where the current status information includes a hardware configuration of the terminal, current load of the operating system of the terminal, resource overheads for cold starting up the application, and duration corresponding to each of a plurality of tasks in a process of cold starting up the application; determining, by analyzing the current status information, a plurality of objects that need to be optimized in the current process of cold starting up the application; and then obtaining, based on the determined objects, a plurality of optimization policies corresponding to these objects, to execute the plurality of optimization policies in the process of cold starting up the application, so as to accelerate the cold startup of the application. The cold startup is accelerated by using the plurality of optimization policies from at least one of two aspects: adjusting resource allocation of the application and accelerating a task in the cold startup process.

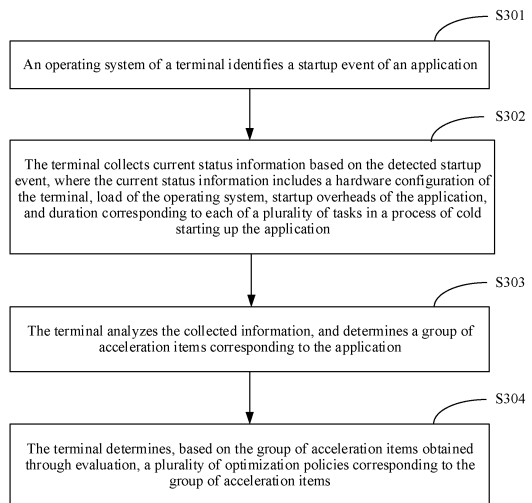


FIG. 3

EP 3 851 956 A1

Description**TECHNICAL FIELD**

[0001] The present invention relates to the computer field, and in particular, to a method, an apparatus, and a terminal for cold starting up an application.

BACKGROUND

[0002] With popularization of smart terminals, more and more applications are installed on the terminals, and a user usually opens a plurality of applications and switches between the plurality of applications. In an Android system, there are three startup modes, and cold startup of an application is a comparatively time-consuming startup mode. The cold startup of the application means that when the application is started up, there is no processes used to run the application in an operating system, in other words, in a startup process of the application, the operating system creates a new process and allocates the new process to the application.

[0003] In the foregoing cold startup process, various resources need to be consumed by the process used to start up the application and the operating system that run the application. These resources may include a processor, for example, a central processing unit (Central Processing Unit, CPU) or a graphics processing unit (Graph Processing Unit, GPU), a memory, an I/O (input/output) resource, and a network resource. Because a plurality of applications are installed on the terminal, a resource allocation manner affects a startup time of the application.

[0004] In an existing application startup method, impact of resource allocation on a startup time of an application is considered. For example, CPU resource allocation is considered in a CPU frequency boost solution of Qualcomm, and memory allocation and I/O resource allocation are considered in a low memory killer (Low Memory Killer) process solution of an Android (Android) system of Google (Google). However, these solutions have an undesirable effect of shortening a time of cold starting up an application, and even in some application scenarios, these solutions may slow down the cold startup of the application. Therefore, the terminal needs more time to display an interface of the application on a screen.

SUMMARY

[0005] In view of this, embodiments of the present invention provide a method, an apparatus, and a terminal for cold starting up an application. A group of optimization policies is more accurately and pertinently determined by analyzing information in a plurality of dimensions such as a using scenario of the terminal, current load of an operating system of the terminal, and system overheads for cold starting up the application. By using these optimization policies together, the terminal can complete the

cold startup of the application more quickly.

[0006] According to a first aspect, an embodiment of the present invention provides a method for cold starting up an application. The method includes: collecting, by a terminal, current status information based on a detected startup event, where the startup event instructs an operating system of the terminal to cold start up an application, and the current status information includes a hardware configuration of the terminal, current load of the operating system, resource overheads for cold starting up the application, and duration corresponding to each of a plurality of tasks in a process of cold starting up the application; determining, based on the current status information, a group of acceleration items corresponding to the application, where the group of acceleration items indicate a plurality of objects that need to be optimized in the current process of cold starting up the application; determining a plurality of optimization policies corresponding to the group of acceleration items, where the plurality of optimization policies include at least one of a first optimization policy and a second optimization policy, the first optimization policy instructs to adjust a resource allocated by the operating system to the cold start, and the second optimization policy is used to accelerate a task in the cold start; and executing the plurality of optimization policies in the process of cold starting up the application.

[0007] In this way, the acceleration items in the cold startup process is determined by analyzing information in the plurality of dimensions such as the using scenario of the terminal, the current load of the operating system of the terminal, and system overheads for cold starting up the application, so that the group of optimization policies is more accurately and pertinently determined based on the acceleration items. By using these optimization policies together, the process of cold starting up the application can be optimized more comprehensively. Therefore, the terminal can complete the cold startup of the application more quickly. In other words, terminals with different hardware configurations, operating systems in different working statuses, and different applications may use different optimization policies. Such optimization can more effectively accelerate the cold startup of the application.

[0008] The plurality of objects include at least one of a resource and a task. The resource includes a hardware resource and a software resource (in other words, how to use the hardware resource). The hardware resource includes at least one of a processing resource, a storage resource, and a network resource. The software resource is used to indicate management on a process by the operating system, for example, a time when a process is allowed to use how many hardware resources. The software resource includes management on another active (active) process by the operating system, for example, at least one of freezing or clearing another active process, releasing memory occupied by the another active process, and releasing a lock occupied by the another process, to allocate the memory and the lock to the process

of cold starting up the application.

[0009] In an implementation, the hardware configuration of the terminal includes specifications of a processor (for example, at least one of a central processing unit, a graphics processing unit, and a coprocessor), a memory device, a network device, and a memory device of the terminal.

[0010] When the terminal cold starts up the application for the first time, both the resource overheads for cold starting up the application and respective duration for performing a plurality of events and a plurality of operations in the process of cold starting up the application are preset values; or when the terminal cold starts up the application for the first time, both the resource overheads for cold starting up the application and respective duration for performing a plurality of events and a plurality of operations in the process of cold starting up the application are historical values.

[0011] This is a process in which the operating system analyzes the collected current status information. The analyzing may be specifically performed based on judgment logic and a threshold that are stored in the system. A correspondence between the evaluation information and an acceleration item may be stored in at least one of a framework layer and a kernel library layer, and the correspondence between the evaluation information and the acceleration item may be that information of each dimension is corresponding to a group of acceleration items at a different level.

[0012] In a possible implementation, the determining, based on the current status information, a group of acceleration items includes: analyzing the current status information, to obtain evaluation information currently corresponding to the application, where the evaluation information currently corresponding to the application includes a level of the hardware configuration of the terminal, a degree of the current load of the operating system, a type of the resource overheads of the application, and a time-consuming task in the cold start; and determining, based on the evaluation information currently corresponding to the application and a correspondence between the evaluation information and an acceleration item, the group of acceleration items currently corresponding to the application.

[0013] This is a process of parsing the collected current status information.

[0014] For the hardware configuration, the terminal is determined as a high-end machine, a mid-range machine, or a low-end machine according to a specific hardware specification. For the load of the operating system, a load level of the operating system such as light, moderate, and heavy, or level-1, level-2, level-3, and level-4 may be determined based on collected information of the load of the operating system.

[0015] It should be understood that in a process of evaluating the load of the operating system, usage of one or more resources by the operating system, namely, load of a plurality of hardware resources and software re-

sources may be evaluated first, and then, the load level of the operating system may be evaluated based on the evaluated load levels of the resources. In another implementation, the load level of the operating system is directly determined based on the collected current status information.

[0016] In a possible implementation, the determining, based on the evaluation information currently corresponding to the application and a correspondence between the evaluation information and an acceleration item, the group of acceleration items currently corresponding to the application includes: when the hardware configuration of the terminal is a first level, the operating system is currently in first level load, the application is of a first resource overheads type, and the time-consuming task in the cold startup is a first group of tasks, currently corresponding, by the application, to a first group of acceleration items; or when the hardware configuration of the terminal is a second level, the operating system is currently in second level load, the application is of a second resource overheads type, and the time-consuming task in the cold startup is a second group of tasks, currently corresponding, by the application, to a second group of acceleration items.

[0017] Two in at least one pair of the first level and the second level, the first level load and the second level load, the first resource overheads type and the second resource overheads type, and the first group of tasks and the second group of tasks are different. For example, the first level and the second level may be different levels, and two in any of the other three pairs have a same level. Of course, two in any of the four pairs may also be different. The first group of acceleration items and the second group of acceleration items are two different groups of acceleration items. In other words, in some scenarios, when hardware levels and/or load levels are different, obtained acceleration items are different even if a same application is cold started up.

[0018] In other words, when two in at least one pair of information in the current status information are analyzed to be different levels, different acceleration items are corresponded.

[0019] In a possible implementation, the determining a plurality of optimization policies corresponding to the group of acceleration items includes: determining, based on the group of acceleration items currently corresponding to the application and a correspondence between the acceleration item and an optimization policy, the plurality of optimization policies corresponding to the group of acceleration items.

[0020] The terminal stores the correspondence between the acceleration item and the optimization policy, and the correspondence may be specifically represented by using a plurality of tables, a pointer, a linked list, or the like.

[0021] In a possible implementation, the current load of the operating system indicates current usage of a hardware resource of the terminal by the operating system,

and the hardware resource includes at least one of a processing resource, a storage resource, and a network resource; and the resource overheads for cold starting up the application indicate usage of at least one of a processor, a memory, a disk, and network bandwidth of the terminal in the cold start.

[0022] In a possible implementation, in the process of cold starting up the application, the plurality of tasks includes at least one first task and at least one second task, and the duration corresponding to each of the plurality of tasks includes respective duration of executing the at least one first task and respective duration of waiting for executing the at least one second task.

[0023] In a possible implementation, the first optimization policy includes adjusting the at least one of the processing resource, the storage resource, and the network resource that are allocated to the cold start.

[0024] In a possible implementation, the first optimization policy includes: boosting an operating frequency of a CPU of the terminal, prolonging a time of the CPU frequency boost of the terminal, and adjusting an energy consumption parameter of the CPU of the terminal, and migrating the process of cold starting up the application to a kernel of another CPU for execution, releasing memory corresponding to the cold startup process of the application in advance, freezing or clearing at least one another process, and adjusting read/write bandwidth or network bandwidth that is allocated to the at least one another process, where the at least one another process is a process corresponding to at least one application, other than the application, that is run in the operating system; and the second optimization policy includes at least one of the following optimization policies: enabling a Nagle algorithm on a TCP connection corresponding to the application, preferentially using the processing resource by a comparatively important thread in the process corresponding to the cold start, reading a file page required for cold starting up the application in advance, performing class verification in advance, and decoding a picture in advance, and loading a basic database in advance.

[0025] It can be learned that after a plurality of types of current status information are comprehensively considered, the group of optimization policies based on the current operating system and features of the application are obtained. These optimization policies include scheduling the resource and accelerating a specific task, and are more targeted and comprehensive.

[0026] According to a second aspect, this application records an apparatus for cold starting up an application. The apparatus includes: a collection module, where the collection module is configured to collect current status information based on a detected startup event, the startup event instructs an operating system of the terminal to cold start up an application, and the current status information includes a hardware configuration of the terminal, current load of the operating system, resource overheads for cold starting up the application, and duration corresponding to each of a plurality of tasks in a

process of cold starting up the application; an analysis module, where the analysis module is configured to, based on the current status information, determine a group of acceleration items corresponding to the application, and the group of acceleration items indicate a plurality of objects that need to be optimized in the current process of cold starting up the application; an optimization policy determining module, where the optimization policy determining module is configured to determine a plurality of optimization policies corresponding to the group of acceleration items, the plurality of optimization policies include at least one of a first optimization policy and a second optimization policy, the first optimization policy instructs to adjust a resource allocated by the operating system to cold start, and the second optimization policy is used to accelerate a task in the cold start; and an execution module, where the execution module is configured to execute the plurality of optimization policies in the process of cold starting up the application.

[0027] According to a third aspect, an embodiment of the present invention records a device, where the device includes a physical circuit, a communications interface, and a storage medium. The storage medium stores a protocol stack program, the communications interface is configured to send and/or receive a data packet with another device by executing the protocol stack program, and the processor is configured to run an instruction in the storage medium, so as to implement the method for cold starting up the application in the first aspect and various implementations of the first aspect.

[0028] In an implementation, the device is a terminal.

[0029] It should be understood that the second aspect and the third aspect respectively describe the apparatus and the device corresponding to the first aspect. For specific implementations, descriptions, and technical effects of the second aspect and the third aspect, refer to the first aspect. Details are not described herein again.

[0030] According to a fourth aspect, a computer program product is provided. The computer program product stores program code that can be used to implement the method in any one of the implementations of the first aspect.

[0031] According to a fifth aspect, a computer readable storage medium is provided, including an instruction. When the instruction is run on a computer, the computer is enabled to perform the method in any one of the implementations of the first aspect.

BRIEF DESCRIPTION OF DRAWINGS

[0032] To describe the technical solutions in the embodiments of the present invention more clearly, the following briefly describes the accompanying drawings required for describing the embodiments. Apparently, the accompanying drawings in the following description show merely some embodiments of the present invention, and persons of ordinary skill in the art may derive other drawings from these accompanying drawings with-

out creative efforts.

FIG. 1 is a schematic diagram of a terminal architecture according to an embodiment of the present invention;

FIG. 2 is a schematic diagram of a software architecture of a terminal according to an embodiment of the present invention;

FIG. 3 is a schematic diagram of a method for cold starting up an application according to an embodiment of the present invention;

FIG. 4 is a schematic diagram of an apparatus for cold starting up an application according to an embodiment of the present invention;

FIG. 5 is a schematic diagram of a device for cold starting up an application according to an embodiment of the present invention; and

FIG. 6 is a schematic structural diagram of a terminal according to an embodiment of the present invention.

DESCRIPTION OF EMBODIMENTS

[0033] Embodiments of the present invention provide a multipath transmission control protocol MPTCP-based packet feedback method, apparatus, and system. The following clearly and completely describes the technical solutions in the embodiments of the present invention with reference to the accompanying drawings in the embodiments of the present invention. All other embodiments obtained by persons of ordinary skill in the art based on the embodiments of the present invention without creative efforts shall fall within the protection scope of the present invention.

[0034] The following describes some terms appearing in this application.

[0035] A and/or B: A and/or B represents A and B, or A or B.

[0036] Kernel mode: A kernel mode is a process running in kernel address space, and the process running in the kernel address space may also be referred to as being in the kernel mode.

[0037] User mode: A user mode is a process running in user address space, and the process running in the user address space may be referred to as being in the user mode.

[0038] Task (task): A task is a basic work unit that is completed by a physical machine in a multi-program or multi-process environment. The task is one or more instruction sequences processed by a control program.

[0039] Running: Running is a type of thread state. It indicates that a thread is running on a CPU.

[0040] Runnable: Runnable is a type of thread state. It indicates that all conditions required for thread running are met, and a thread is queuing in a CPU running queue and is waiting for CPU scheduling, in other words, waiting for using the CPU.

[0041] Uninterruptable sleep (uninterruptible sleep):

Uninterruptible sleep is a type of thread state, and is called sleep for short. It indicates that a thread is in an uninterruptible blocking state. In this state, the thread is waiting for various lock resources.

[0042] WakeKill: WakeKill is a type of thread state. It indicates that a thread is in an uninterruptible blocking state, and is blocked by a critical signal, and is generally waiting for reading or writing a disk.

[0043] Block I/O (input/output): Block I/O (input/output) is an input/output of a disk of a block device, in other words, reading/writing of the disk.

[0044] Inflate: Inflate represents a process of generating a corresponding view (view) based on resource file context and a parameter in an Android system.

[0045] DDR: DDR is double data rate synchronous dynamic random access memory (Double Data Rate Synchronous Dynamic Random Access Memory, DDR SDRAM for short), and is a type of memory.

[0046] VerifyClass: In an Android system, verifyclass is a process in which some classes of an application cannot be verified during compilation due to some reasons, but are verified during application running. For example, after a hot patch function is used in this application, this process easily occurs.

[0047] Blocking: Blocking is to suspend a specific thread to wait for a condition. The condition can be that a specific resource is ready.

[0048] Process killer: Process killer is to end a process or release a resource (such as memory, CPU, I/O, and lock) occupied by a process.

[0049] Cold startup of an application is divided into three phases: preparation before cold startup of an application process, application initialization, and display of an application that is started up on an interface of a terminal. An intuitive description is a process from a time when a user taps a screen or a specific button, or when a user speaks a speech to the terminal, to a time when an application interface is completely displayed to the user.

[0050] In an implementation, the preparation before the cold startup of the application process includes: receiving, by the terminal, a touchscreen event input by the user, and triggering a series of processing. For example, the touch screen event is used to enable an operating system of the terminal to identify that an application is required to be cold started up. The series of processing includes identifying the touchscreen event, where the event is delivered, by using an input daemon process, to a launcher's onClick() function of a launcher progress (Graphical User Interface, GUI), and further includes foreground/background switching. Specifically, the launcher progress requests a system service (System Server) process to perform process initialization, window initialization, and the like on a to-be-started up application, and then the system server switches an application running on the launcher to a background application, and sets the to-be-started up application as a foreground application.

[0051] Application initialization includes initialization of a common component of the application, and initialization of a service and data that are required for running the process of the application. The common component may include at least one of a network request library, a JavaScript object notation (JavaScript Object Notation, Json) parser, a big data statistics library, and an application encryption library.

[0052] The application that is started up is displayed on the terminal interface, including operations such as loading data that needs to be displayed and layout and drawing of the interface, for example, downloading a text, a picture, or a video from a network, decoding a picture, and reading a database in which to-be-displayed data is located.

[0053] The operating system cold starts up the application by using a process. One process includes one or more threads, for example, a component loading thread and a main thread. The cold startup process requires cooperation of threads in the process, so that tasks in the foregoing several phases are executed in a specific sequence, and resources of the terminal need to be consumed. The tasks in the foregoing several phases may affect a time for cold starting up the application. In an existing solution to accelerating the cold startup of the application, several fixed manners are usually used to accelerate the cold start. For example, a CPU frequency boost manner or a manner in which the process is killed for releasing memory is only used. These manners are not used to pertinently optimize performance of the cold startup application, in many cases, may deteriorate performance of the cold startup application.

[0054] A time consumed by cold starting up one application may be referred to as a startup delay. The startup delay is an accumulation of duration corresponding to each of a plurality of tasks in a process of cold starting up the application. Comparatively large impact on the startup delay comes from a delay of executing some time-consuming tasks, and resource (including a software resource and a hardware resource) contention between a plurality of threads of the process. The startup delay may include at least one of a CPU delay of one or more threads, a memory slow path delay, a zram compression delay, an I/O delay, a network delay, a delay of waiting for a lock by a thread in a sleep state (a delay of waiting for a lock in a sleep state for short), a delay of a thread in a runnable state, and a delay of thread blocking of the thread (for example, the component loading thread blocks the main thread) in the process of executing the application.

[0055] CPU delay: A CPU delay is duration during which a thread is waiting for using the CPU, in other words, is duration during which the thread keeps in a runnable state.

[0056] Memory slow path delay: A memory slow path delay is a time consumed by a thread to use memory through a slow path. The memory read by the slow path is memory that is released by a device according to a

read/write instruction and that is occupied by another application or process.

[0057] Zram compression delay: A zram compression delay is a time used by a thread to read compressed data in the memory. Zram, also called zip RAM, is a function of a Linux kernel and may be used for virtual memory compression. To store more data in the memory, there are some data whose address storage is compressed in the memory, and the data needs to be decompressed before being read.

[0058] I/O delay: An I/O delay is a time consumed by a thread to execute a batch of read/write disk instructions. The I/O delay mainly comes from a waiting time of these instructions in a queue. In the cold startup process, data required for cold starting up the application needs to be read, for example, a user interface, user data, and a local cache of the application. Therefore, the CPU may deliver read/write disk instructions in batches. An operating system of the terminal always has read/write disk instructions from all threads, and these instructions are placed in a queue. Because the read/write disk instructions triggered by the cold startup of the application are delivered in batches, it is usually needed to wait in the queue for a period of time before all the instructions are executed. Generally, the I/O delay accounts for about 15% of duration of cold starting up the application.

[0059] The method for cold starting up the application described in this application is applied to a terminal, and the terminal is a device that can be installed with an application and that is provided for a user to use. The terminal obtains, from a plurality of dimensions through current status collection, current status information related to the cold startup of the application, to analyze, based on the collected current status information, a problem of accelerating the cold startup of the application in this specific scenario (in other words, to obtain a group of acceleration items). In addition, an optimization policy corresponding to the determined acceleration items may be obtained based on the determined acceleration items, and a policy for the specific problem is used to optimize the cold startup process of the application, so as to reduce a time of cold starting up the application, and more properly allocate the software resource and the hardware resource of the terminal, thereby improving user experience.

[0060] The terminal may store one or more optimization policy libraries, and each optimization policy library stores a plurality of optimization policies. In terms of the hardware resource, the optimization policy may be adjusting the hardware resource as required. An adjustable manner includes at least one of the following: boosting an operating frequency of a CPU, binding a process that executes the application to a specific CPU core or real-locating a CPU core that is used to execute the application, adjusting an operating frequency of an L3 cache, adjusting an operating frequency of a GPU, adjusting a power management (always-on) parameter, and adjusting both an operating frequency of the CPU and an op-

erating frequency of the DDR to enable the adjusted operating frequency of the CPU and the adjusted operating frequency of the DDR to still match each other. In terms of accelerating the task in the cold start, the optimization policy may include: scheduling a VIP thread if a runnable state of a key thread is long, using page-level file pre-fetching if the I/O delay is long, using a TCP protocol to transmit a packet with no-delay if the network delay is long, performing page fault lock degradation if the uninterruptible sleep is long, and using another process to execute, in advance, a task that takes a long time. The task may be at least one of function compiling, picture decoding, class verifying, So library loading, TCP pre-connecting, and the like.

[0061] The following describes specific meanings of some optimization policies.

[0062] CPU boost: CPU boost is used to improve CPU processing performance. Common technical means are CPU frequency boost and CPU core affinity. The CPU frequency boost is to boost an operating frequency of the CPU to a maximum value or a larger value in an operating frequency range. The CPU core affinity is to bind some applications to some cores with a comparatively high CPU configuration to ensure that these applications can be allocated with more processing resources. It should be understood that the CPU may work within a frequency range. In consideration of a service life and power consumption, generally, the CPU works at an intermediate value or a comparatively small value in the frequency range. In addition, the terminal may have a default CPU frequency boost time, in other words, duration during which an operating system adjusts the CPU to work at a comparatively high frequency after receiving a CPU frequency boost instruction, for example, 1.5s.

[0063] VIP thread scheduling: VIP thread scheduling refers to inserting some threads of threads that are queuing in a processor running queue and waiting for executing by a processor. In other words, the some threads are executed in advance, for example, the some threads are replaced in the front of a queue. Alternatively, VIP thread scheduling refers to switching some threads of threads from one processor core to another processor core for running, and the some threads may be comparatively important threads that are sorted by priority, or may be a specific type or several types of preset threads. The another processor core that is switched to is a processor core with a light load, a comparatively short waiting time, or in an idle state.

[0064] Process fast killer: Process fast killer means that when a KILL_SIG used to terminate a process is received. Memory is released only when a reference count of virtual memory that is shared by a thread of the process in a running state is reduced to 0. In addition, the shared virtual memory is adjusted to memory of an anonymous page and memory of a file page that are first released. The KILL_SIG is a signal that is sent to a process and that terminates the process immediately. The process fast killer can solve a problem of a large memory

pressure, so as to quickly release the memory.

[0065] Page fault lock degradation: A Page fault is a page fault. Page fault lock degradation is a dynamic memory allocation technology. Specifically, page allocation is delayed until a page to be accessed by a process is not in a physical memory, thereby causing a speculative page fault (Speculative Page Fault). For example, in a page fault process (for example, a page fault page fault process), a process-level mm semaphore mm semaphore is no longer used, and instead a thread-level lock with a finer granularity is used. A process of changing from a process-level lock to the thread-level lock is the lock degradation. This optimization policy can provide better parallel performance in the process.

[0066] Page-level file pre-fetching: Page-level file pre-fetching means that a file page that needs to be read in the cold startup process is identified in advance. In the cold startup process, the page is pre-fetched, in advance, from a cache by using another process instead that an entire file in which the pages are located is pre-fetched. Specifically, the optimization policy of the terminal may reduce I/O wait of a startup time.

[0067] Using a TCP protocol to transmit a packet with no-delay (no-delay): Using a TCP protocol to transmit a packet with no-delay (no-delay) is also referred to as transmitting a packet by the TCP with no-delay, and is applied to a scenario in which interaction needs to be performed with a network side by using the TCP protocol or an MPTCP protocol in the cold startup process. Specifically, in the cold startup process of the application, a Nagle algorithm in the TCP protocol is enabled. In this algorithm, a packet transmission delay is small, and there is no need to wait for another packet, thereby saving a packet transmission time.

[0068] Picture pre-decoding, So library pre-loading, and function pre-compiling are all necessary pre-processing that needs to be performed on data in advance in the cold startup process. The pre-processing is performed by another process after the operating system detects a startup event. Generally, when the another process performs the pre-processing, the process used to cold start up the application is still in a construction phase. This manner can ensure that when the process used to cold start up the application needs to use the data, the process does not need to wait for processing or reading the data.

[0069] Picture pre-decoding: Picture pre-decoding means that before cold start, a group of pictures that need to be decoded in the cold startup are identified, and after a startup event is detected, another process is used to decode these pictures. For example, these pictures may be determined based on a historical cold startup process of the application.

[0070] So library pre-loading: So library pre-loading means that before cold start, So library information that needs to be loaded in the cold startup is identified. In the cold start, another thread is used to load the files in advance. The So library stores .so files in an Android sys-

tem.

[0071] Function pre-compiling: Function pre-compiling means that functions or code blocks that take a long time to precompile or that are run frequently in the cold startup process are identified based on a historical cold startup process, and these code blocks are precompiled in advance at the background. Specifically, a JIT compiler in a virtual machine may be used to determine code blocks that are run frequently. These code blocks that are run frequently are also referred to as hot spot codes (hot spot code).

[0072] Background application freezing: Background application freezing means that an application that is still running at the background is frozen by using a freezer mechanism of cgroup in cold startup of an application, in other words, freezing a process that runs the background application, for example, forbidding the process to access memory or occupy CPU and unfreezing the process after the cold startup of the application is complete. This optimization policy can reduce impact of the application that is run at the background on cold startup duration when a system is heavily loaded.

[0073] Application management and control sorting: Application management and control sorting means that process lists that can be managed and controlled at the background are generated based on a priority of services corresponding to a plurality of processes that are run in a current operating system, such as a killable process list, a compressible process list, and a non-killable process list. These process lists can be used by cooperating with the foregoing optimization policies.

[0074] File cache classification mechanism: A file cache classification mechanism is used to classify and manage caches of files used in a cold startup process. The classification management file cache is implemented by classifying least recently used (Least Recently Used, LRU) queues. Specifically, a comparatively important file cache is added in an LRU queue with a higher priority, and the other file caches are added in an LRU queue with a lower priority. When the file cache is deleted, the cache in the LRU queue with a lower priority is first deleted.

[0075] The foregoing plurality of optimization policies are merely examples, and optimization policies that may be used in the method described in this application are not limited to the foregoing types.

[0076] The following describes in detail the method for cold starting up an application from a plurality of aspects.

[0077] In this application, the terminal is a physical machine, and may also be referred to as user equipment (User Equipment, "UE" for short), a mobile station (Mobile Station, "MS" for short), a mobile terminal (Mobile Terminal), or the like. Optionally, the terminal may have a capability of communicating with one or more core networks by using a radio access network (Radio Access Network, RAN). For example, the terminal may be a mobile phone (or referred to as a "cellular" phone), or a computer with a mobile property. For example, the terminal

may alternatively be a portable, pocket-sized, handheld, computer built-in, or in-vehicle mobile apparatus, such as a mobile phone, a tablet computer, a notebook computer, a palmtop computer, an intelligent wearable device, or a mobile Internet device (English: mobile internet device, MID for short). It should be understood that, in addition to the terminal, the method provided in the embodiments of the present invention may be further applied to another type of computer system.

[0078] Load of the operating system of the terminal may also be referred to as a running state of the operating system, and indicates usage of a hardware resource of the terminal by software of the terminal. Current load of the operating system of the terminal includes current load of the operating system and resource overheads of an application that is currently run in the operating system. The hardware resource includes but is not limited to at least one of a CPU, a memory (for example, a DDR), a disk, a network, and a GPU. For example, a process of the application reads data from the memory, the disk, or a network by using the CPU or the GPU. The hardware resource and how to use the hardware resource may be described by using a plurality of parameters. This is not limited in this application. For example, the parameters related to the disk are a capacity of a local disk, a read/write rate local disk, I/O wait (I/O wait) of a local disk, and the like, and the parameters related to the network are network bandwidth, a network delay, and the like.

[0079] In one implementation, the load of the operating system is described at least from usage of three resources: the CPU, the memory, and the disk. The terminal may determine a status of the operating system based on real-time data of the three resources used by the operating system. Each of the foregoing three aspects may be described by using different parameters, for example, CPU load, available memory, and I/O wait. The CPU load refers to utilization of the CPU, and the available memory may be an available memory amount, or may be a proportion of the available memory amount to a total memory amount. For another example, the I/O wait may be described by using duration of the I/O wait, or may be described by using a quantity of parallel I/O channels. In a Linux-based operating system, there is a dedicated instruction for obtaining values of the foregoing three parameters. Certainly, the load of the operating system may alternatively be described by using a parameter related to another resource. This is not limited in this application. For example, if a to-be-cold started-up application needs to download data from a network device side, a network resource such as a network signal strength, network bandwidth, and a current data transmission status of the terminal need to be further considered. The terminal may access the network by using an access node of a wireless network or an access node of a wired network, for example, may be an access point (Access Point, AP) of a wireless Wi-Fi network, an Ethernet router, an access point of a Winmax network, a base station of a wireless cellular

mobile network, and the like. A type of the network and a specific form of the access node are not limited in the present invention.

[0080] After analyzing the obtained values of the foregoing three parameters, the operating system may determine usage of the current system in terms of the CPU, the memory, and the disk, so as to determine the status of the operating system. An analysis result may be performed by comparing with a preset threshold. For example, if CPU load exceeds a first threshold, it is considered that CPU load is high; if CPU load is between a first threshold and a second threshold, it is considered that the CPU load is moderate; and if available memory is less than a specific proportion, it is considered that available memory is low.

[0081] In an implementation, the status of the operating system is classified into several levels, for example, three levels, namely, light, moderate, and heavy. The operating system may directly determine the status of the operating system based on the obtained values of the foregoing three parameters, or may determine the status of the operating system based on the levels corresponding to the values of the three parameters. For example, an operating system with high CPU load, low available memory, and high I/O wait is in a heavy load state.

[0082] In an implementation, resource overheads for cold starting up the application are also described based on usage of three resources: the CPU, the memory, and the disk. The resource overheads also referred to as startup overheads of the application for short, namely, processing resource overheads (for example, CPU overheads), memory overheads, and I/O overheads. All the three aspects may be represented by using different parameters. The CPU overheads are CPU resources occupied by the process to execute the cold startup application, for example, may be represented by a percentage of the CPU resources to total CPU resources. The memory overheads are memory used by the process to execute the cold startup application, and may be represented, for example, by a percentage of the memory to total memory, or may be represented by a used memory amount. The I/O overheads are an amount of disk data that needs to be read or written by the process to execute the cold startup application. The operating system may also store some thresholds, so that the operating system analyzes values of the foregoing three types of overheads. For example, if the CPU overheads exceed a threshold, it is considered that CPU overheads of the cold startup process of the application is high. It should be understood that the startup overheads of the application may also be indicated by using a combination of other resources, for example, at least one of a processor, a memory, a disk, or network bandwidth.

[0083] FIG. 1 is a schematic structural diagram of a terminal 100 according to an embodiment of the present invention. The terminal 100 may be a device in this embodiment of the present invention. Some modules that may be included at a software layer and a hardware layer

of the terminal are schematically drawn in the figure. The software layer of the terminal includes a driver layer, a kernel library layer, a framework layer, and an application layer. A CPU driver, a GPU driver, a display controller driver, and the like are performed at the driver layer. A kernel of an operating system (for example, a kernel of an Android system) is also located at the driver layer. It should be understood that FIG. 1 is merely a possible schematic layer division, and some layers may have different names in division in another manner. For example, in another schematic diagram, FIG. 2 shows another division manner. The driver layer in FIG. 1 is referred to as a Linux kernel in FIG. 2, and the Linux kernel includes driver described at the driver layer in FIG. 1.

[0084] The kernel library layer is a core part of the operating system, and includes an appearance manager, a media framework, a relational database, a 2G graphics engine library, a web browser engine, a kernel library, a virtual machine (for example, Dalvik Virtual Machine), and the like. In addition, the framework layer may include a browser engine, a layout engine, a file parser, and the like. The application layer may include a plurality of application programs such as a home (home) screen, a media player (Media Player), and a browser (Browser).

[0085] The hardware layer of the terminal includes a central processing unit (Center Processing Unit, CPU), a graphics processing unit (Graphic Processing Unit, GPU), and the like, and certainly, may further include a memory, an input/output device, a memory, a memory controller, a network interface, and the like. The input device may include a keyboard, a mouse, a touchscreen, and the like. The output device may include a display device such as a liquid crystal display (Liquid Crystal Display, LCD), a cathode ray tube (Cathode Ray Tube, CRT), holographic imaging (Holographic) device, or a projector (Projector). An operating system (such as Android or Firefox OS) and some application programs may run above the hardware layer.

[0086] The system architecture shown in FIG. 1 may be used to perform the method described in this application, and FIG. 2 describes locations of various subsystems that may be used to perform the method described in this application in another system architecture. This system architecture is similar to the system architecture described in FIG. 1. FIG. 2 includes a software layer of a terminal. The software layer includes a Linux kernel (kernel), a kernel library layer, a framework layer, and an application layer. In FIG. 2, a double-headed arrow is used to represent data interaction. An intra-application scheduling subsystem, a real-time processing subsystem, a collection execution adaptation subsystem, a kernel extension subsystem, and a non-real-time processing subsystem shown in FIG. 2 are improvement parts of an existing terminal operating system related in this application in some implementations.

[0087] An application process is used to run an application. One application process relates to an application layer and a framework layer. If one application is run in

a virtual machine, the application process further relates to a kernel library layer. Parts of one application process at different layers exchange information by using an interface. FIG. 2 shows three application processes in an operating system, and shows parts, at three layers, of an application process 1 that is on a left side and that is run in the virtual machine. The application process 1 executes service logic of the application at the application layer. The intra-application scheduling subsystem includes a part in the framework layer and a part in the virtual machine, and may manage a plurality of applications in the system. The management includes implementing collection, identification, recording, and storage (namely, intra-application high time-consuming collection and execution in the figure) of an intra-application time-consuming condition and implementing an optimization policy that belongs to intra-application scheduling (the optimization policy is stored in an intra-application scheduling solution set). The application process 1 is run on the virtual machine, and therefore the intra-application scheduling subsystem shown in FIG. 2 is distributed at both the framework layer and a virtual machine layer. The intra-application scheduling subsystem is in the solution set of the framework layer, and the optimization policy includes but is not limited to picture pre-decoding, class (View) pre-verifying (inflate), TCP pre-connecting, and So library pre-loading. In the solution set of the framework layer, the optimization policy includes but is not limited to: function compiling in advance and class loading in advance.

[0088] The framework layer includes the real-time processing subsystem and the non-real-time processing subsystem. The real-time processing subsystem is configured to implement functions such as onsite collection, overheads evaluation, policy formulation, and event identification. These functions are also focuses of the application cold startup method described in this application, and are further described in detail below with reference to a method procedure. For example, the onsite collection function is corresponding to a collection module 401 in FIG. 4, and the overheads evaluation and the policy formulation are corresponding to an analysis module 402 in FIG. 4. A module corresponding to the event recognition is not shown in FIG. 4. The subsystem also includes a real-time solution set, and the solution set records some improvement solutions that belong to the framework layer, for use of formulating an overall improvement policy. For example, a solution in the solution set may include: accelerating the CPU as required, and if the application that needs to be cold started up uses a large amount of memory, changing a memory allocation policy, to ensure memory use of the application.

[0089] The non-real-time processing subsystem stores a configuration and data related to the cold startup of the application, and the data may be stored in a disk, for example, storing historical data applied to cold startup in different system load statuses, a default configuration of each sub-solution, a default parameter value, and an

optimized parameter value obtained after the sub-solution is used. In an implementation process of a processing subsystem formulation policy, the configuration and the data may be obtained from the non-real-time processing subsystem.

[0090] The core layer includes a collection execution adaptation subsystem and the part of the application process that belongs to the core layer. The collection execution adaptation subsystem is configured to process a policy formulated by the real-time processing subsystem, to enable the kernel extension subsystem to execute the policy. In an implementation, the collection execution adaptation subsystem has comparatively high permission. The collection execution adaptation subsystem includes an execution current status collection module, configured to provide a query interface for a current status collection module of an internship subsystem to use; a policy delivery module, configured to receive a command of a real-time subsystem policy formulating module, and deliver the command to a kernel subsystem; and a platform adaptation module, configured to adapt to encapsulate a heterogeneous kernel node, for example, an interface of a resource such as a CPU and a GPU, into a unified interface, for invoking by the real-time processing subsystem.

[0091] The driver layer includes the kernel extension subsystem, and the kernel extension subsystem is configured to, in the cold startup process of the application, execute policies formulated by the real-time processing subsystem. These policies may include one of CPU-or DDR-or GPU-related scheduling, VIP thread scheduling, process fast killer, lock (such as PageFault) degrading, page-level file pre-fetching, low memory killer (LMK), and using a TCP protocol to transmit a packet with no-delay (no-delay).

[0092] It can be learned that the operating system maintains an analysis rule in which an acceleration item is analyzed based on the current status information. The operating system further maintains a mapping relationship between the acceleration item and the optimization policy. The mapping relationship may include at least one of a mapping relationship between a group of the acceleration items and a group of optimization policies and a mapping relationship between one acceleration item and at least one optimization policy. The determined optimization policy may be a group of optimization policies directly obtained by using the group of the acceleration items. Alternatively, at least one optimization policy corresponding to each obtained acceleration item may be obtained for the obtained acceleration item, and all the obtained optimization policies are optimization policies to be executed in the cold startup of the application. Alternatively, at least one optimization policy corresponding to each obtained acceleration item may be obtained for the obtained acceleration item, and according to a combination rule, some optimization policies are removed from all the obtained optimization policies, to obtain an optimization policy to be executed in the cold star-

tup of the application.

[0093] In this way, the device shown in FIG. 2 may be configured to perform the application cold startup method recorded in this application. It can be learned that program code corresponding to the method in this application exists in operating system platform software. During running, program code of the present invention is run in a host memory of a server.

[0094] With reference to FIG. 3, the following describes, by using a mobile phone as an example, an application cold startup method recorded in this application. For explanations of various terms in the method, refer to corresponding paragraphs in this application. The embodiment corresponding to FIG. 3 is merely an example for description.

[0095] S301. An operating system of a terminal identifies a startup event of an application, where the startup event instructs the operating system of the terminal to cold start up the application.

[0096] For example, the event may be a tap performed by a user on a screen of the terminal, or a voice instruction sent by a user to the terminal, or an image received by the terminal, or a signal or a software instruction that can be recognized by the operating system of the terminal. A manner of starting up the event is not limited in this application.

[0097] S302. The terminal collects current status information based on a detected startup event, where the current status information includes a hardware configuration of the terminal, load of the operating system, startup overheads of the application, and duration corresponding to each of a plurality of tasks in a process of cold starting up the application.

[0098] Resource overheads for cold starting up the application and the duration corresponding to each of the plurality of tasks in the process of cold starting up the application used in the method procedure described in this application are not real-time values of the cold start. The resource overheads and the duration may be values that are of a cold startup process of the application and that are historically recorded by the terminal, or may be default values stored by the terminal, or may be estimated values of the operating system of the terminal. It should be understood that the startup overheads of the application vary with the load of the system. For example, in a high-load system state and a low-load system state, parameter values of startup overheads of a same application may be different. In an implementation, the terminal stores startup load values of a same application in different statuses of the operating system, so that a more appropriate value is used to describe the startup load in a specific status of the operating system. In one cold start, the terminal may collect various parameters of startup load in the current cold startup process, and record a status of the operating system in the current cold start, for use of cold starting up the application in the future.

[0099] The hardware configuration of the terminal includes a plurality of hardware specifications of the terminal,

and the plurality of hardware includes at least one of a CPU, a DDR, a GPU, a memory, and a network resource. Hardware specifications can be represented by a plurality of parameters, such as a CPU frequency, a quantity of CPU cores, a DDR bandwidth frequency, a GPU frequency, a quantity of GPU cores, a memory and disk read/write speed, and a memory capacity.

[0100] The load of the operating system may be represented by three parameters of the system: processing, storage, and network, and may include at least one of CPU load, available memory, I/O wait, and a network state of the system.

[0101] The startup overheads of the application are also referred to as overheads for cold starting up the application, and the overheads include at least one of an average time, computing power MIPS, CPU load, GPU load, occupied memory, an I/O read/write amount, I/O read/write bandwidth, and a network reading amount for starting up each application.

[0102] When the application is not cold started up for the first time, duration corresponding to a plurality of tasks in the process is determined based on historical data for starting up the application on the mobile phone. The duration corresponding to the task may be duration for executing a specific task, for example, duration for executing any one of tasks such as class verifying, function compiling, picture decoding, So library loading, and TCP link establishing during running, or may be duration for waiting for executing a specific task, for example, at least one of a delay of waiting for the CPU, a memory slow path delay, a zram compression delay, an I/O delay, and a network delay of a main thread and another key path. For the first cold start, a preset value or an estimated value may be used.

[0103] S303. The terminal parses the collected information, and determines a group of acceleration items corresponding to the application.

[0104] In an implementation, the step includes: analyzing the current status information, to obtain evaluation information currently corresponding to the application, where the evaluation information currently corresponding to the application includes a level of the hardware configuration of the terminal, a degree of the current load of the operating system, a type of the resource overheads of the application, and a time-consuming task in the cold start; and determining, based on the evaluation information currently corresponding to the application and a correspondence between the evaluation information and an acceleration item, the group of acceleration items currently corresponding to the application.

[0105] The determining the acceleration items includes performing overheads evaluation, in other words, analyzing the current status information, to obtain evaluation information currently corresponding to the application.

[0106] The overheads evaluation includes evaluating levels of the hardware configuration, the load of the operating system, the startup overheads of the application,

and a scheduling delay and a running time consumption of the application. The evaluation may include classifying various types of information and determining an event that affects a cold startup delay of the application. The following evaluation may be determined based on a pre-set threshold.

[0107] For the hardware configuration, the terminal is determined as a high-end machine, a mid-range machine, or a low-end machine according to a specific hardware specification.

[0108] For the load of the operating system, a load level of the operating system such as light, moderate, and heavy, or level-1, level-2, level-3, and level-4 may be determined based on collected information of the load of the operating system.

[0109] It should be understood that in a process of evaluating the load of the operating system, usage of one or more resources by the operating system, namely, load of a plurality of hardware resources and software resources may be evaluated first, and then, the load level of the operating system may be evaluated based on the evaluated load levels of the resources.

[0110] In another implementation, the load level of the operating system is directly determined based on the collected current status information.

[0111] It can be learned that the terminal stores a rule used to evaluate the load of the operating system. For example, if a CPU is heavily loaded, available memory is low, I/O is high, and available network bandwidth is small, the system is heavily loaded. Alternatively, if the CPU load is greater than a first threshold, and the available memory is less than a second threshold, the system is heavily loaded.

[0112] For the startup overheads of the application, that the application is of at least one of a CPU consumption type, a GPU consumption type, a memory consumption type, an I/O consumption type, and a network consumption type is determined based on resource consumption when the application is started up.

[0113] Identifying a time-consuming task for starting up the application is specifically identifying, based on historical data for starting up the application on the mobile phone, a key bottleneck of starting up the application in terms of a resource scheduling delay and an application running delay. In terms of the resource scheduling delay, the application is determined as at least one of a running queue waiting type, a memory allocation delay type, an I/O delay type, or the like based on historical scheduling delay information for starting up the application in a condition of current resource load (which may be represented by a load level) of the terminal. In terms of the application running delay, the application is determined as at least one of a function compiling time-consuming type, a picture decoding time-consuming type, a layout time-consuming type, a class verifying time-consuming type, a network waiting time-consuming type, or the like based on collected application startup time-consuming execution information.

[0114] The overheads evaluation further includes obtaining a corresponding group of acceleration items based on a result of the foregoing analysis. These acceleration items are used to indicate at least one of a resource or a task that needs to be adjusted in this cold start. Specifically, several items with a largest resource gap estimated for current startup of the application are evaluated, according to a resource gap ranking, based on identified hardware, identified load of the operating system, and identified overheads of a to-be-started-up application of the mobile phone. Based on historical data of the to-be-started-up application on the mobile phone, several tasks with a largest scheduling gap are estimated and several tasks that need to be optimized during the cold startup according to a time ratio of the scheduling delay.

[0115] In an implementation, when the hardware configuration of the terminal is a first level, the operating system is currently in first level load, the application is of a first resource overheads type, and the time-consuming task in the cold startup is a first group of tasks, the application is currently corresponding to a first group of acceleration items; or when the hardware configuration of the terminal is a second level, the operating system is currently in second level load, the application is of a second resource overheads type, and the time-consuming task in the cold startup is a second group of tasks, the application is currently corresponding to a second group of acceleration items.

[0116] S304. The terminal determines, based on the group of acceleration items obtained through evaluation, a plurality of optimization policies corresponding to the group of acceleration items.

[0117] The terminal stores the correspondence between the acceleration item and the optimization policy, and the correspondence may be represented by using a table or a plurality of tables, or may be represented by using a pointer or a linked list. The correspondence may be stored in at least one of a framework layer, a Linux kernel, and a kernel library layer. The correspondence may be a correspondence between a plurality of groups of acceleration items and a plurality of groups of optimization policies. One group of acceleration items are corresponding to one group of optimization policies. Alternatively, the correspondence may be a correspondence between a plurality of acceleration items and a plurality of optimization policies, and one acceleration item are corresponding to one or more optimization policies. In this way, all optimization policies corresponding to the group of acceleration items may be obtained by searching the group of acceleration items obtained through evaluation one by one. In this manner, operations such as necessary deduplication of same optimization policies and elimination of mutually opposite optimization policies may be further included. The opposite optimization policies have opposite optimization directions. For example, an optimization policy 1 instructs the CPU to boost a frequency, and an optimization policy 2 instructs the CPU

to reduce a frequency. These two optimization policies are opposite.

[0118] The plurality of optimization policies include at least one of a first optimization policy and a second optimization policy, where the first optimization policy instructs to adjust a resource allocated by the operating system to the cold start, and the second optimization policy is used to accelerate a task in the cold start.

[0119] Specifically, the determined optimization policy may be invoked from an existing library, or may be obtained by modifying, based on collected current status information, a parameter in an existing policy.

[0120] For descriptions of a determining manner of the optimization policy and a type of the optimization policy, refer to the foregoing related paragraphs. Only a simple example is used herein for description.

[0121] In terms of the resource, the optimization policy may be adjusting a parameter of the hardware resource as required. The adjustable parameter includes at least one of the following: CPU frequency boost, CPU core affinity, a frequency modulator parameter, an L3 cache frequency, and the like; a GPU frequency and a power management (always-on) parameter; and a DDR frequency and an association between the DDR and the CPU.

[0122] In terms of the task, the optimization policy may include: scheduling a VIP thread when a runnable state of a key thread is long; using page-level cache pre-fetching if the I/O delay is long; using TCP to transmit a packet with no-delay if the network delay is long; and performing page fault lock degradation when an uninterruptible sleep state is long. The optimization policy further includes asynchronously executing a comparatively time-consuming startup process in advance. The startup process may be at least one of hotspot function compiling, picture decoding, class verifying, So library loading, TCP pre-connecting, and the like.

[0123] It can be learned that S302 to S304 are corresponding to current status collection, overheads evaluation, event identification, and policy formulation in the real-time processing subsystem in FIG. 2. Certainly, the real-time processing subsystem may further include a real-time system sub-solution set, and the real-time system sub-solution set is configured to schedule each optimization policy from a service side of the framework layer. The sub-solution set includes but is not limited to accelerating the CPU as required, providing memory assurance for starting up a memory-bound application, and the like. In addition, a process from S302 to S304 also involves obtaining the historical data, a default configuration, and the like from the non-real-time processing subsystem.

[0124] S305. The terminal executes the determined optimization policies in the process of cold starting up the application.

[0125] Specifically, the real-time processing subsystem in FIG. 2 may deliver, by using a collection execution adaptation subsystem, instructions corresponding to the

optimization policies to a kernel extension subsystem, so as to execute the optimization policies.

[0126] In this way, the terminal obtains, from a plurality of dimensions by collecting current status information, software and hardware configurations related to the cold startup of the application, to analyze, based on the collected information, a problem of accelerating the cold startup of the application in this specific scenario. In addition, a policy for the specific problem is used to optimize the cold startup process of the application, so as to reduce a time of cold starting up the application, and more properly allocate the software resource and the hardware resource of the terminal, thereby improving user experience.

[0127] It can be learned that in the method described in this application, the hardware configuration of the terminal affects a cold startup policy applied to the terminal. The following uses two specific scenarios as examples for description. In one scenario, an application on a terminal A is cold started up, and a hardware configuration of the terminal A is comparatively low. In the other scenario, an application on a terminal B is cold started up, and a hardware configuration of the terminal B is comparatively high.

[0128] For the terminal A, the terminal first collects current status information, including a hardware configuration, load of the operating system, resource overheads for cold starting up the application, and duration corresponding to each of a plurality of tasks in a process of cold starting up the application. The hardware configuration may be obtained by an operating system of the terminal A by using some hardware configuration collection means, for example, testing or sensing by using a sensor or an interface, or may be obtained by reading a hardware configuration pre-stored in the terminal A.

[0129] The operating system of the terminal A obtains the following hardware configuration: A CPU is a dual-core chip and a specific model is Qualcomm 8917, a memory of the terminal A is 2 GB, a GPU is a 308 chip, and a flash memory is a Hynix universal flash storage (Universal Flash Storage, UFS). It can be learned that the terminal A is a mobile phone with a comparatively low hardware configuration. A main bottleneck of the terminal A lies in a low CPU configuration, and processing resources are easily insufficient. The terminal A uses an operating system of the Android 8.2 version. Because the hardware configuration is comparatively low, after the terminal A is powered on, even in a standby state, the operating system of the terminal A is moderately loaded.

[0130] After a user uses three or four applications, the operating system detects that CPU load of the terminal A remains at about 60%, available memory of the terminal A is about 300 MB, and I/O wait of the terminal A remains at 10%. Generally, the I/O wait that is higher than 4% belongs to high I/O wait. In other words, the operating system of the terminal A is in a state of high CPU load, low available memory, and high I/O wait, and it may be

considered that the operating system is heavily loaded. In other words, when the operating system is heavily loaded, the terminal detects an instruction of the user, and the instruction instructs the terminal to start up a camera application.

[0131] In addition, the terminal determines, based on the status of the operating system, that resource overheads for cold starting up the camera application are 30% of CPU overheads (overhead), 30 MB of I/O overheads (to-be-read disk data), and 800 MB of memory overheads. In a process of starting up the camera application, a memory slow path delay is 1200 ms, an I/O delay is 600 ms, a delay of a running application of a thread in a runnable state is 500 ms, a delay of waiting for a lock in a sleep state is 120 ms, and a delay of blocking of a main thread by a component loading thread is 300 ms.

[0132] Then, the terminal A evaluates, based on the collected information, a main problem of limiting a cold startup speed. Based on the collected information, the terminal A analyzes that a main cause of a weak hardware capability of the terminal A is that a CPU processing capability is insufficient. That the system of the terminal A is heavily loaded is reflected in a low available memory rate and a high I/O utilization because of the application that is run. Startup overheads of the camera application are high memory overheads, high CPU overheads, and general I/O overheads. A startup delay of the camera application mainly comes from the memory slow path delay, the I/O delay, the delay of the running application of the thread in the runnable state, and the delay of waiting for the lock in the sleep state.

[0133] Then, a plurality of corresponding optimization policies are determined based on an evaluation result.

[0134] For example, in terms of the hardware resource, the optimization policy includes prolonging a boost time of the CPU to 5s. This is because a default boost time of the CPU of the terminal A is 1.5s, a CPU configuration is low, and the processing resources are insufficient. In terms of reducing the startup delay, the optimization policy includes: reducing a memory allocation time, reducing a runtime time, and reducing the I/O delay. For example, in terms of reducing the memory allocation time, at least one of the following optimization policies may be executed, including: allocating a memory fast killer interface to the camera application in the cold startup process of the camera application, calculating in advance a memory amount that needs to be killed, and determining that more memory is killable, the killed memory is no longer occupied by another application, and in a startup phase, a background application is prohibited from being restarted up after being killed; and in terms of reducing the runtime time, a background application of the operating system can be frozen for a period of time (for example, a period of time equal to a CPU boost time of cold starting up the application), and the freezing of the background application means that when the camera is cold started up, the background application cannot use CPU resources temporarily; or a priority of scheduling a CPU of an initializa-

tion thread of a component of the camera application may be increased. In terms of reducing the I/O delay, a file for starting up the camera application may be pre-read.

[0135] After the optimization policies are determined, the terminal may execute the optimization policies in the cold startup process of the camera, to accelerate the cold start.

[0136] In this way, the terminal obtains, from a plurality of dimensions by collecting current status information, software and hardware configurations related to the cold startup of the application, to analyze, based on the collected information, a problem of accelerating the cold startup of the application in this specific scenario. In addition, a policy for the specific problem is used to optimize the cold startup process of the application, so as to reduce a time of cold starting up the application, and more properly allocate the software resource and the hardware resource of the terminal, thereby improving user experience. For example, in the foregoing scenario in which the terminal A cold starts up the camera application, if the foregoing manner is not used to optimize the cold startup process, a time for cold starting up the camera application is 5975 ms. If the foregoing manner is used, a time for cold starting up the camera application is 3885 ms.

[0137] The following uses another specific scenario as an example for description. A hardware configuration of a terminal B used in this scenario is comparatively high, and an application that needs to be cold started up is an e-commerce application, for example, Taobao (taobao) or Amazon (Amazon) used for online shopping. Similarly, the terminal first collects current status information.

[0138] An operating system of the terminal B obtains the following hardware configuration: A CPU is an 8-core chip and a specific model is Kirin 970, a memory of the terminal B is 6 GB, and a flash memory is Toshiba UFS2.0. It can be learned that the terminal A is a mobile phone with a comparatively high hardware configuration, and only a flash memory configuration is slightly insufficient. The terminal A uses an operating system of the Android 8.2 version. Because the hardware configuration is comparatively high, the operating system of the terminal B used by most users is lightly loaded or moderately loaded.

[0139] The operating system of the terminal B detects that CPU load of the terminal B is at about 10%, available memory of the terminal B is about 4 GB, I/O wait of the terminal B is almost 0%, and a network state of the terminal B is good. The operating system of the terminal B is lightly loaded. In other words, when the operating system is lightly loaded, the terminal detects an instruction of a user, and the instruction instructs the terminal to cold start up the e-commerce application.

[0140] In addition, the terminal determines, based on the status of the operating system, that application load of the camera application is 30% of CPU overheads (overhead), I/O overheads, where the I/O overheads include 100 MB of disk data that needs to be read and 1

MB of disk data that needs to be written, and 259 MB of memory overheads. In a process of cold starting up the application, a network delay is 200 ms, an I/O delay is 500 ms, a delay of just-in-time compiler (Just in Time, JIT) of some functions in the application is 500 ms, and a delay of waiting for a lock by a thread in a sleep state is 100 ms, a delay of verification class (Verifyclass) tasks such as user identity verification is 200 ms, and a delay of picture decoding is 60 ms.

[0141] Then, the terminal B evaluates, based on the collected information, a main problem of limiting a cold startup speed. Based on the foregoing collected information, the terminal B analyzes that a hardware capability of the terminal B is strong, the operating system is lightly loaded, and there are sufficient resources to cold start up the application. The application is an application whose cold startup overheads are general, and a startup delay of the application mainly comes from the network delay, the I/O delay, and the delay of just-in-time compiler of the function.

[0142] Then, a plurality of corresponding optimization policies are determined based on an evaluation result. A hardware resource scheduling parameter does not need to be adjusted, and a default policy is used. For example, frequency boost of 1.5 seconds is maintained in both the cold startup and interface switching. In terms of the scheduling delay, for a long I/O delay, page-level file pre-fetching is performed; for a long network delay, TCP packet transmission with no-delay is performed; and for a long sleep time, a page fault lock degradation solution is used. In terms of intra-application invoking, for time-consuming JIT, an ahead-of-time compiler speed-profile of a hotspot function is used; for a long process of the verification class, asynchronous verification class is performed in advance; and for time-consuming decoding, asynchronous decoding is performed in advance.

[0143] After the optimization policies are determined, the terminal may execute the policies in the cold startup process of the application, to accelerate the cold start.

[0144] In this way, the terminal analyzes, based on the current status information, a problem of accelerating the cold startup of the application in this specific scenario. In addition, a policy for the specific problem is used to optimize the cold startup process of the application, so as to reduce a time of cold starting up the application, and more properly allocate the software resource and the hardware resource of the terminal, thereby improving user experience. For example, in the foregoing scenario, if the foregoing manner is not used to optimize the cold startup process, a time for cold starting up the e-commerce application is 3534 ms. If the foregoing manner is used, a time for cold starting up the e-commerce application is 2163 ms.

[0145] FIG. 4 describes an apparatus 400 for cold starting up an application according to this application. The apparatus 400 includes a collection module 401, where the collection module 401 is configured to collect current status information based on a detected startup event, the

startup event instructs an operating system of the terminal to cold start up an application, and the current status information includes a hardware configuration of the terminal, current load of the operating system, resource overheads for cold starting up the application, and duration corresponding to each of a plurality of tasks in a process of cold starting up the application; an analysis module 402, where the analysis module 402 is configured to, based on the current status information, determine a group of acceleration items corresponding to the application, the group of acceleration items indicate a plurality of objects that need to be optimized in the current process of cold starting up the application; an optimization policy determining module 403, where the optimization policy determining module 403 is configured to determine a plurality of optimization policies corresponding to the group of acceleration items, the plurality of optimization policies include at least one of a first optimization policy and a second optimization policy, the first optimization policy instructs to adjust a resource allocated by the operating system to cold start, and the second optimization policy is used to accelerate a task in the cold start; and an execution module 404, where the execution module 404 is configured to execute the plurality of optimization policies in the process of cold starting up the application.

[0146] For specific implementation details of the apparatus, refer to the foregoing description of the method. In an implementation, the collection module 401 in FIG. 4 may be implemented at the framework layer and the kernel library layer in FIG. 2. For example, an instruction for collecting the current status information may be delivered at the framework layer, the hardware configuration of the terminal, the current load of the operating system, the resource overheads for cold starting up the application, and duration corresponding to a part of tasks in the cold startup application process in the current status information are all generated at the kernel library layer, and duration corresponding to the other part of tasks in the cold startup application process may be generated at the framework layer. The analysis module 402 and the optimization policy determining module 403 may also be implemented at the framework layer in FIG. 2, and the execution module 404 may be implemented at a Linux kernel in FIG. 2.

[0147] In this way, the acceleration items in the cold startup process may be determined by analyzing information in a plurality of dimensions such as the using scenario of the terminal, the current load of the operating system of the terminal, and system overheads for cold starting up the application, so that the group of optimization policies is more accurately and pertinently determined based on the acceleration items, thereby optimizing the process of cold starting up the application. Therefore, the terminal can complete the cold startup of the application more quickly.

[0148] In an implementation, in the process of cold starting up the application, the plurality of tasks includes at least one first task and at least one second task, and

the duration corresponding to each of the plurality of tasks includes respective duration of executing the at least one first task and respective duration of waiting for executing the at least one second task. For example, the first task may be at least one of picture decoding, class verifying, So file loading, and function compiling that are mentioned above, and the second task may be at least one of waiting for using a CPU, zram compressing, and disk data reading/writing.

[0149] In an implementation, the analysis module 402 is specifically configured to: analyze the current status information, to obtain evaluation information currently corresponding to the application, where the evaluation information currently corresponding to the application includes a level of the hardware configuration of the terminal, a degree of the current load of the operating system, a type of the resource overheads of the application, and a time-consuming task in the cold start; and determine, based on the evaluation information currently corresponding to the application and a correspondence between the evaluation information and an acceleration item, the group of acceleration items currently corresponding to the application.

[0150] This is a process in which the operating system analyzes the collected current status information. The analyzing may be specifically performed based on judgment logic and a threshold that are stored in the system. A correspondence between the evaluation information and an acceleration item may be stored in at least one of a framework layer and a kernel library layer, and the correspondence between the evaluation information and the acceleration item may be that information of each dimension is corresponding to a group of acceleration items at a different level.

[0151] In an implementation, in an aspect of determining, based on the evaluation information currently corresponding to the application and a correspondence between the evaluation information and an acceleration item, the group of acceleration items currently corresponding to the application, the analysis module 402 is further configured to: when the hardware configuration of the terminal is a first level, the operating system is currently in first level load, the application is of a first resource overheads type, and the time-consuming task in the cold startup is a first group of tasks, currently correspond, by the application, to a first group of acceleration items; or when the hardware configuration of the terminal is a second level, the operating system is currently in second level load, the application is of a second resource overheads type, and the time-consuming task in the cold startup is a second group of tasks, currently correspond, by the application, to a second group of acceleration items.

[0152] Two in at least one pair of the first level and the second level, the first level load and the second level load, the first resource overheads type and the second resource overheads type, and the first group of tasks and the second group of tasks are different. For example, the

first level and the second level may be different levels, and two in any of the other three pairs have a same level. Of course, two in any of the four pairs may also be different. The first group of acceleration items and the second group of acceleration items are two different groups of acceleration items. In other words, in some scenarios, when hardware levels and/or load levels are different, obtained acceleration items are different even if a same application is cold started up.

[0153] In other words, when two in at least one pair of information in the current status information are analyzed to be different levels, different acceleration items are corresponded.

[0154] Certainly, when levels of two in each of the foregoing four pairs of information are not completely consistent, acceleration items corresponding to the application are all a same group of acceleration items such as the first group of acceleration items.

[0155] In an implementation, the optimization policy determining module 403 is specifically configured to, based on the group of acceleration items currently corresponding to the application and a correspondence between the acceleration item and an optimization policy, determine the plurality of optimization policies corresponding to the group of acceleration items.

[0156] For details, refer to the foregoing paragraphs.

[0157] In an implementation, the current load of the operating system indicates current usage of a hardware resource of the terminal by the operating system, and the hardware resource includes at least one of a processing resource, a storage resource, and a network resource; and the resource overheads for cold starting up the application indicate usage of at least one of a processor, a memory, a disk, and network bandwidth of the terminal in the cold start.

[0158] In an implementation, the first optimization policy includes adjusting the at least one of the processing resource, the storage resource, and the network resource that are allocated to the cold start.

[0159] In an implementation, the first optimization policy includes: boosting an operating frequency of a CPU of the terminal, prolonging a time of the CPU frequency boost of the terminal, and adjusting an energy consumption parameter of the CPU of the terminal, and migrating the process of cold starting up the application to a kernel of another CPU for execution, releasing memory corresponding to the cold startup process of the application in advance, freezing or clearing at least one another process, and adjusting read/write bandwidth or network bandwidth that is allocated to the at least one another process, where the at least one another process is a process corresponding to at least one application, other than the application, that is run in the operating system; and the second optimization policy includes at least one of the following optimization policies: enabling a Nagle algorithm on a TCP connection corresponding to the application, preferentially using the processing resource by a comparatively important thread in the process corresponding

to the cold start, reading a file page required for cold starting up the application in advance, performing class verification in advance, and decoding a picture in advance, and loading a basic database in advance.

[0160] FIG. 5 describes a structure of a device 500 for cold starting up an application according to this application. The device 500 includes at least one processing circuit 501, a storage medium 502, and a communications interface 504. The device 500 may include at least one network interface and at least one communications bus 605. The communications bus 605 is configured to implement connection and communication between these components. The device 50 optionally includes a user interface, and includes a display (for example, a touchscreen, an LCD, a CRT, holographic (Holographic) imaging, or a projector (Projector)), a keyboard, or a click device (for example, a mouse, a trackball (trackball), a touchpad, or a touchscreen).

[0161] The storage medium 502 may include a read-only memory and a random access memory, and provide an instruction and data to the processing circuit 501. A part of the storage medium 502 may further include a nonvolatile random access memory (NVRAM).

[0162] In some implementation manners, the storage medium 502 stores the following elements: an executable module or a data structure, or a subset thereof, or an extended set thereof.

[0163] The operating system 5021 includes various system programs, for example, the framework layer, the kernel library layer, and the driver layer shown in FIG. 1, or the framework layer, the kernel library layer, and the Linux kernel shown in FIG. 2. The various system programs are configured to implement basic services and process hardware-based tasks. In an implementation, the collection module 401, the analysis module 402, the optimization policy determining module 403, and the execution module 404 mentioned above are all included in the operating system 5021.

[0164] The application program module 5022 includes various application programs, for example, the application that is cold started up described in this application, or for example, the gallery, the media player (Media Player), and the browser (Browser) shown in FIG. 1. The various system programs configured to implement various application services.

[0165] In this embodiment of the present invention, by invoking a program or an instruction stored in the storage medium 502, the processing circuit 501 collects current status information based on a detected startup event, the startup event instructs an operating system of the terminal to cold start up an application, and the current status information includes a hardware configuration of the terminal, current load of the operating system, resource overheads for cold starting up the application, and duration corresponding to each of a plurality of tasks in a process of cold starting up the application; determines, based on the current status information, a group of acceleration items corresponding to the application, where

the group of acceleration items indicate a plurality of objects that need to be optimized in the current process of cold starting up the application; determines a plurality of optimization policies corresponding to the group of acceleration items, where the plurality of optimization policies include at least one of a first optimization policy and a second optimization policy, the first optimization policy instructs to adjust a resource allocated by the operating system to cold start, and the second optimization policy is used to accelerate a task in the cold start; and executes the plurality of optimization policies in the process of cold starting up the application.

[0166] The device corresponding to FIG. 5 invokes the program or the instruction stored in the storage medium 502, and the processor 501 may perform any possible implementation of the method and the apparatus described above. Details are not described herein again. The hardware at the hardware layer described in FIG. 1 may also be considered as a specific implementation of FIG. 5. For example, the processing circuit 501 in FIG. 5 is represented as a central processing unit and a graphics processing unit in FIG. 1, and the storage medium 502 is represented as a memory in FIG. 1.

[0167] In this way, acceleration items in the cold startup process may be determined by analyzing information in a plurality of dimensions such as the using scenario of the terminal, the current load of the operating system of the terminal, and system overheads for cold starting up the application, so that the group of optimization policies is more accurately and pertinently determined based on the acceleration items. By using these optimization policies together, the process of cold starting up the application can be optimized more comprehensively. Therefore, the terminal can complete the cold startup of the application more quickly.

[0168] FIG. 6 is a schematic structural diagram of a terminal according to an embodiment of the present invention. The terminal may be configured to perform some or all of the steps of the application cold startup method described above. For example, reference may be made to related descriptions, specific descriptions, and descriptions of beneficial effects that are corresponding to FIG. 3. Details are not described herein again. As shown in FIG. 6, the terminal 600 includes components such as a radio frequency (Radio Frequency, RF) circuit 601, a memory 602, an input unit 603, a display unit 604, a sensor 605, an audio circuit 606, a wireless fidelity (wireless fidelity, Wi-Fi) module 607, a processor 608, and a power supply 609. Persons skilled in the art may understand that, the terminal structure shown in FIG. 6 does not constitute a limitation on a mobile phone, and the terminal may include more or fewer components than those shown in the diagram, or some components may be combined, or the components may be arranged in a different manner.

[0169] FIG. 6 may be understood as a specific implementation based on FIG. 5. There is a correspondence between some components shown in the two figures. For

example, the memory 602 in FIG. 6 is corresponding to the storage medium 502 in FIG. 5, and the processor 608 in FIG. 6 is corresponding to the processing circuit 501 in FIG. 5. The radio frequency (Radio Frequency, RF) circuit 601 and the wireless fidelity (wireless fidelity, Wi-Fi) module 607 in FIG. 6 are specific implementations of the communications interface 504 in FIG. 5. Therefore, the application layer, the framework layer, the kernel layer, and the driver layer in FIG. 1, the various layers and the various subsystems in FIG. 2, and the various modules in FIG. 4 may also be implemented by the processor 608 by invoking code in the memory 602. Certainly, the terminal corresponding to FIG. 6 describes more hardware components than those in FIG. 5.

[0170] The RF circuit 601 may be configured to receive and send a signal in an information sending and/or receiving process or a call process. In particular, after receiving downlink information from a base station, the RF circuit 610 sends the downlink information to the processor 608 for processing. In addition, the RF circuit 910 sends designed uplink data to the base station. Usually, the RF circuit 601 includes but is not limited to an antenna, at least one amplifier, a transceiver, a coupler, a low noise amplifier (Low Noise Amplifier, LNA), a duplexer, and the like. In addition, the RF circuit 601 may further communicate with a network and another device through wireless communication. Any communications standard or protocol may be used in the wireless communication, including but not limited to a global system for mobile communications (Global System of Mobile communication, GSM), a general packet radio service (General Packet Radio Service, GPRS), code division multiple access (Code Division Multiple Access, CDMA), wideband code division multiple access (Wideband Code Division Multiple Access, WCDMA), long term evolution (Long Term Evolution, LTE), an email, a short message service (Short Messaging Service, SMS), and the like.

[0171] The memory 602 stores a computer program and the computer program includes an application program 6021 and an operating system program 6022. The processor 608 is configured to read the computer program in the memory 602, and then execute a method defined in the computer program. For example, the processor 608 reads the operating system program 6022, so as to run an operating system on the terminal 600 and implement various functions of the operating system. Alternatively, the processor 608 reads one or more application programs 6021, so as to run an application on the terminal 600. The operating system 6022 includes a computer program that can implement the memory reclamation method provided in this embodiment of the present invention, so that after the processor 608 reads the operating system program 6022 and runs the operating system, the operating system may possess a memory reclamation function provided in the embodiments of the present invention. In addition, the memory 602 further stores other data 6023 different from the computer program, where the other data 6023 may include data gen-

erated after the operating system 6022 or the application program 6021 is run, and the data includes system data (for example, configuration parameters of the operating system) and user data. In addition, the memory 602 generally includes a memory and an external memory. The memory may be a random access memory (RAM), a read-only memory (ROM), a cache (CACHE), or the like. The external memory may be a hard disk, an optical disk, a USB, a floppy disk, a tape drive, or the like. The computer program is usually stored in the external memory. Before performing processing, the processor loads the computer program from the external memory to the memory.

[0172] The input unit 603 may be configured to receive input digit or character information, and generate a key signal input related to a user setting and function control of the terminal 600.

[0173] Specifically, the input unit 603 may include a touch panel 6031 and another input device 6032. The touch panel 6031, also referred to as a touchscreen, may collect a touch operation (for example, an operation performed by a user on the touch panel 6031 or near the touch panel 6031 by using any proper object or accessory such as a finger or a stylus) performed by the user on or near the touch panel 951, and drive a corresponding connection apparatus based on a preset program. Optionally, the touch panel 6031 may include two parts: a touch detection apparatus and a touch controller. The touch detection apparatus detects a touch direction of the user, detects a signal brought by the touch operation, and transmits the signal to the touch controller. The touch controller receives touch information from the touch detection apparatus, converts the touch information into touch coordinates, then sends the touch coordinates to the processor 608, and can receive and execute a command sent by the processor 608. In addition, the touch panel may be implemented by using a plurality of types such as a resistive type, a capacitive type, an infrared ray, and a surface acoustic wave type 6031. The input unit 603 may include the another input device 6032 in addition to the touch panel 6031. Specifically, the another input device 6032 may include but is not limited to one or more of a physical keyboard, a function key (for example, a volume control key or an on/off key), a trackball, a mouse, a joystick, and the like.

[0174] A display unit 604 may be configured to display information entered by a user, information provided to the user, and various menus of the mobile phone. The display unit 604 may include a display panel 6041. Optionally, the display panel 6041 may be configured in a form of a liquid crystal display (Liquid Crystal Display, LCD), an organic light-emitting diode (Organic Light-Emitting Diode, OLED), or the like. Further, the touch panel 6031 may cover the display panel 6041. When detecting a touch operation on or near the touch panel 6031, the touch panel 6031 transmits the touch operation to the processor 608 to determine a type of a touch event, and then the processor 608 provides a corresponding

visual output on the display panel 6041 based on the type of the touch event. In FIG. 6, the touch control panel 6031 and the display panel 6041 are used as two independent components to implement input and output functions of the mobile phone. However, in some embodiments, the touch panel 6031 and the display panel 6041 may be integrated to implement the input and output functions of the mobile phone.

[0175] The sensor 605 may be a light sensor, a motion sensor, or another sensor. Specifically, the optical sensor may include an ambient light sensor and a proximity sensor. The ambient light sensor may adjust luminance of the display panel 6041 based on brightness of ambient light, and when the mobile phone approaches an ear, the proximity sensor may turn off the display panel 6041 and/or backlight. As a type of motion sensor, an accelerometer sensor may detect a value of acceleration in each direction (usually on three axes), may detect a value and a direction of gravity in a stationary state, and may be used in an application for identifying a mobile phone posture (such as screen switching between a landscape mode and a portrait mode, a related game, or magnetometer posture calibration), a function related to vibration identification (such as a pedometer or a knock), or the like. Other sensors such as a gyroscope, a barometer, a hygrometer, a thermometer, or an infrared sensor may be further configured in the mobile phone. Details are not described herein again.

[0176] For example, the startup event described in this application may be a user touch display unit 604, and the pressure signal is transmitted by the sensor 605 to the terminal.

[0177] An audio frequency circuit 606, a loudspeaker 6061, and a microphone 6062 may provide an audio interface between the user and the mobile phone. The audio frequency circuit 606 may convert received audio data into an electrical signal, and transmit the electrical signal to the loudspeaker 6061, and the loudspeaker 6061 converts the electrical signal into a sound signal for output. In addition, the microphone 6062 converts a collected sound signal into an electrical signal, and the audio frequency circuit 606 receives the electrical signal, converts the electrical signal into audio data, and outputs the audio data to the processor 608 for processing, so as to send the audio data to, for example, another mobile phone by using the RF circuit 601, or output the audio data to the memory 602 for further processing.

[0178] Wi-Fi is a short-range wireless transmission technology, and the terminal may help, by using the Wi-Fi module 607, the user to send and/or receive an e-mail, browse a web page, access streaming media, and the like. The Wi-Fi module 607 provides wireless broadband internet access for the user. Although FIG. 6 shows the Wi-Fi module 607, it can be understood that the Wi-Fi module 670 is not a necessary constituent of the terminal and may be omitted well as required provided that the essence of the present invention is not changed.

[0179] The processor 608 is a control center of the ter-

minal, and connects all parts of the entire terminal by using various interfaces and lines. By running or executing a software program and/or a module stored in the memory 602 and invoking data stored in the memory 602, the processor 608 performs various functions of the terminal and data processing, to perform overall monitoring on the terminal. Optionally, the processor 608 may include one or more processors. For example, the processor 608 may include one or more central processing units, or include one central processing unit and one graphics processing unit. When the processor 608 includes a plurality of processors, the plurality of processors may be integrated in a same chip, or may be chips separate from each other. One processor may include one or more processing cores.

[0180] The terminal 600 further includes the power supply 609 (for example, a battery) that supplies power to the components. Preferably, the power supply may be logically connected to the processor 608 by using a power supply management system. In this way, functions such as management of charging, discharging, and power consumption are implemented by using the power supply management system.

[0181] Although not shown in the figure, the terminal may further include a camera, a Bluetooth module, and the like. Details are not described herein again.

[0182] In the foregoing embodiment, the method procedure of each step and the apparatus 400 corresponding to FIG. 4 may also be implemented based on a structure of the terminal. All in FIG. 1 may be considered as a component of an abstract structure of the processor 608.

[0183] In this embodiment of the present invention, the processor 608 is configured to perform the following operations by invoking program code stored in the memory 602:

collecting, by a terminal, current status information based on a detected startup event, the startup event instructs an operating system of the terminal to cold start up an application, and the current status information includes a hardware configuration of the terminal, current load of the operating system, resource overheads for cold starting up the application, and duration corresponding to each of a plurality of tasks in a process of cold starting up the application; determining, based on the current status information, a group of acceleration items corresponding to the application, where the group of acceleration items indicate a plurality of objects that need to be optimized in the current process of cold starting up the application; determining a plurality of optimization policies corresponding to the group of acceleration items, where the plurality of optimization policies include at least one of a first optimization policy and a second optimization policy, the first optimization policy instructs to adjust a resource allocated by the operating system to cold start, and the second optimization policy is used to accelerate a task in the cold start; and executing the plurality of optimization policies in the process of cold starting up the

application.

[0184] The device corresponding to FIG. 6 invokes the program or the instruction stored in the memory 602, and the processor 608 may perform any possible implementation of the method and the apparatus described above. Details are not described herein again.

[0185] In this way, the acceleration items in the cold startup process may be determined by analyzing information in a plurality of dimensions such as the using scenario of the terminal, the current load of the operating system of the terminal, and system overheads for cold starting up an application, so that the group of optimization policies is more accurately and pertinently determined based on the acceleration items. By using these optimization policies together, the process of cold starting up the application can be optimized more comprehensively. Therefore, the terminal can complete the cold startup of the application more quickly.

[0186] It should be noted that, to make the description brief, the foregoing method embodiments are expressed as a series of actions. However, persons skilled in the art should appreciate that the present invention is not limited to the described action sequence, because according to the present invention, some steps may be performed in other sequences or performed simultaneously. In addition, persons skilled in the art should also appreciate that all the embodiments described in the specification are example embodiments, and the related actions and modules are not necessarily mandatory to the present invention.

[0187] In the foregoing embodiments, the description of each embodiment has respective focuses. For a part that is not described in detail in an embodiment, refer to related descriptions in other embodiments.

[0188] In the several embodiments provided in this application, it should be understood that the disclosed apparatus may be implemented in other manners. For example, the described apparatus embodiment is merely an example. For example, the module division is merely logical function division and may be other division in actual implementation. For example, a plurality of modules or components may be combined or integrated into another system, or some features may be ignored or may not be performed. In addition, the displayed or discussed mutual couplings or direct couplings or communication connections may be implemented through some interfaces. The indirect couplings or communication connections between the apparatuses or modules may be implemented in electrical or other forms.

[0189] The modules described as separate parts may or may not be physically separate, and parts displayed as modules may or may not be physical modules, may be located in one position, or may be distributed on a plurality of network modules. Some or all the modules may be selected according to actual needs to achieve the objectives of the solutions of the embodiments.

[0190] In addition, functional modules in the embodiments of the present invention may be integrated into

one processing module, or each of the modules may exist alone physically, or two or more modules are integrated into one module. The integrated module may be implemented in a form of hardware, or may be implemented in a form of a software functional module.

[0191] When the integrated module is implemented in the form of a software functional module and sold or used as an independent product, the integrated unit may be stored in a computer readable memory. Based on such an understanding, the technical solutions of the present invention essentially, or the part contributing to the prior art, or all or some of the technical solutions may be implemented in the form of a software product. The software product is stored in a memory and includes several instructions for instructing a computer device (which may be a personal computer, a server, or a network device) to perform all or some of the steps of the methods described in the embodiments of the present invention. The foregoing storage medium includes: any medium that can store program code, such as a USB flash drive, a read-only memory (ROM, Read-Only Memory), a random access memory (RAM, Random Access Memory), a removable hard disk, a magnetic disk, or an optical disc.

[0192] Persons of ordinary skill in the art may understand that all or some of the steps of the methods in the embodiments may be implemented by a program instructing relevant hardware. The program may be stored in a computer readable storage medium. The storage medium may include a flash memory, a read-only memory (Read-Only Memory, ROM for short), a random access memory (Random Access Memory, RAM for short), a magnetic disk, and an optical disc.

[0193] The embodiments of the present invention are described in detail above. In this specification, the principle and implementation of the present invention are described herein through specific examples. The description about the embodiments is merely provided to help understand the method of the present invention. In addition, persons of ordinary skill in the art can make variations and modifications to the present invention in terms of the specific implementations and application scopes according to the content recorded in the application file. Therefore, the content of this specification shall not be construed as a limit to the present invention.

Claims

1. A method for cold starting up an application, wherein the method comprises:

collecting, by a terminal, current status information based on a detected startup event, wherein the startup event instructs an operating system of the terminal to cold start up an application, and the current status information comprises a hardware configuration of the terminal, current load of the operating system, resource over-

heads for cold starting up the application, and duration corresponding to each of a plurality of tasks in a process of cold starting up the application;

determining, based on the current status information, a group of acceleration items corresponding to the application, wherein the group of acceleration items indicate a plurality of objects that need to be optimized in the current process of cold starting up the application; determining a plurality of optimization policies corresponding to the group of acceleration items, wherein the plurality of optimization policies comprise at least one of a first optimization policy and a second optimization policy, the first optimization policy instructs to adjust a resource allocated by the operating system to the cold start, and the second optimization policy is used to accelerate a task in the cold start; and executing the plurality of optimization policies in the process of cold starting up the application.

- 2. The method according to claim 1, wherein the determining, based on the current status information, a group of acceleration items comprises:

analyzing the current status information, to obtain evaluation information currently corresponding to the application, wherein the evaluation information currently corresponding to the application comprises a level of the hardware configuration of the terminal, a degree of the current load of the operating system, a type of the resource overheads of the application, and a time-consuming task in the cold start; and determining, based on the evaluation information currently corresponding to the application and a correspondence between the evaluation information and an acceleration item, the group of acceleration items currently corresponding to the application.

- 3. The method according to claim 2, wherein the determining, based on the evaluation information currently corresponding to the application and a correspondence between the evaluation information and an acceleration item, the group of acceleration items currently corresponding to the application comprises:

when the hardware configuration of the terminal is a first level, the operating system is currently in first level load, the application is of a first resource overheads type, and the time-consuming task in the cold startup is a first group of tasks, currently corresponding, by the application, to a first group of acceleration items; or when the hardware configuration of the terminal

is a second level, the operating system is currently in second level load, the application is of a second resource overheads type, and the time-consuming task in the cold startup is a second group of tasks, currently corresponding, by the application, to a second group of acceleration items.

5

10

15

20

25

30

35

40

45

50

55

- 4. The method according to any one of claims 1 to 3, wherein the determining a plurality of optimization policies corresponding to the group of acceleration items comprises:

determining, based on the group of acceleration items currently corresponding to the application and a correspondence between the acceleration item and an optimization policy, the plurality of optimization policies corresponding to the group of acceleration items.

- 5. The method according to any one of claims 1 to 4, wherein the current load of the operating system indicates current usage of a hardware resource of the terminal by the operating system, and the hardware resource comprises at least one of a processing resource, a storage resource, and a network resource; and the resource overheads for cold starting up the application indicate usage of at least one of a processor, a memory, a disk, and network bandwidth of the terminal in the cold start.

- 6. The method according to any one of claims 1 to 5, wherein in the process of cold starting up the application, the plurality of tasks comprise at least one first task and at least one second task, and the duration corresponding to each of the plurality of tasks comprises respective duration of executing the at least one first task and respective duration of waiting for executing the at least one second task.

- 7. The method according to any one of claims 1 to 6, wherein the first optimization policy comprises adjusting the at least one of the processing resource, the storage resource, and the network resource that are allocated to the cold start.

- 8. The method according to any one of claims 1 to 6, wherein the first optimization policy comprises: boosting an operating frequency of a CPU of the terminal, prolonging a time of the CPU frequency boost of the terminal, and adjusting an energy consumption parameter of the CPU of the terminal, and migrating the process of cold starting up the application to a kernel of another CPU for execution, releasing memory corresponding to the cold startup process of the application in advance, freezing or clearing at least one another process, and adjusting read/write bandwidth or network bandwidth that is allocated to the at least one another process, wherein the at least

one another process is a process of at least one application, other than the application, that is run in the operating system; and
 the second optimization policy comprises at least one of the following optimization policies: enabling a Nagle algorithm on a TCP connection corresponding to the application, preferentially using the processing resource by a comparatively important thread in the process corresponding to the cold start, reading a file page required for cold starting up the application in advance, performing class verification in advance, and decoding a picture in advance, and loading a basic database in advance.

- 9. An apparatus for cold starting up an application, wherein the apparatus comprises:

a collection module, wherein the collection module is configured to collect current status information based on a detected startup event, the startup event instructs an operating system of the terminal to cold start up an application, and the current status information comprises a hardware configuration of the terminal, current load of the operating system, resource overheads for cold starting up the application, and duration corresponding to each of a plurality of tasks in a process of cold starting up the application;
 an analysis module, wherein the analysis module is configured to, based on the current status information, determine a group of acceleration items corresponding to the application, and the group of acceleration items indicate a plurality of objects that need to be optimized in the current process of cold starting up the application;
 an optimization policy determining module, wherein the optimization policy determining module is configured to determine a plurality of optimization policies corresponding to the group of acceleration items, the plurality of optimization policies comprise at least one of a first optimization policy and a second optimization policy, the first optimization policy instructs to adjust a resource allocated by the operating system to cold start, and the second optimization policy is used to accelerate a task in the cold start; and
 an execution module, wherein the execution module is configured to execute the plurality of optimization policies in the process of cold starting up the application.

- 10. The apparatus according to claim 9, wherein the analysis module is specifically configured to: analyze the current status information, to obtain evaluation information currently corresponding to the application, wherein the evaluation information currently corresponding to the application comprises a level of the hardware configuration of the terminal, a de-

gree of the current load of the operating system, a type of the resource overheads of the application, and a time-consuming task in the cold start; and determine, based on the evaluation information currently corresponding to the application and a correspondence between the evaluation information and an acceleration item, the group of acceleration items currently corresponding to the application.

- 11. The apparatus according to claim 10, wherein in an aspect of determining, based on the evaluation information currently corresponding to the application and a correspondence between the evaluation information and an acceleration item, the group of acceleration items currently corresponding to the application, the analysis module is specifically configured to: when the hardware configuration of the terminal is a first level, the operating system is currently in first level load, the application is of a first resource overheads type, and the time-consuming task in the cold startup is a first group of tasks, currently correspond, by the application, to a first group of acceleration items; or when the hardware configuration of the terminal is a second level, the operating system is currently in second level load, the application is of a second resource overheads type, and the time-consuming task in the cold startup is a second group of tasks, currently correspond, by the application, to a second group of acceleration items.
- 12. The apparatus according to any one of claims 9 to 11, wherein the optimization policy determining module is specifically configured to, based on the group of acceleration items currently corresponding to the application and a correspondence between the acceleration item and an optimization policy, determine the plurality of optimization policies corresponding to the group of acceleration items.
- 13. The apparatus according to any one of claims 9 to 12, wherein the current load of the operating system indicates current usage of a hardware resource of the terminal by the operating system, and the hardware resource comprises at least one of a processing resource, a storage resource, and a network resource; and the resource overheads for cold starting up the application indicate usage of at least one of a processor, a memory, a disk, and network bandwidth of the terminal in the cold start.
- 14. The apparatus according to any one of claims 9 to 13, wherein in the process of cold starting up the application, the plurality of tasks comprise at least one first task and at least one second task, and the duration corresponding to each of the plurality of tasks comprises respective duration of executing the at least one first task and respective duration of waiting for executing the at least one second task.

15. The apparatus according to any one of claims 9 to 14, wherein the first optimization policy comprises adjusting the at least one of the processing resource, the storage resource, and the network resource that are allocated to the cold start. 5

16. The apparatus according to any one of claims 9 to 15, wherein the first optimization policy comprises: boosting an operating frequency of a CPU of the terminal, prolonging a time of the CPU frequency boost of the terminal, and adjusting an energy consumption parameter of the CPU of the terminal, and migrating the process of cold starting up the application to a kernel of another CPU for execution, releasing memory corresponding to the cold startup process of the application in advance, freezing or clearing at least one another process, and adjusting read/write bandwidth or network bandwidth that is allocated to the at least one another process, wherein the at least one another process is a process corresponding to at least one application, other than the application, that is run in the operating system; and the second optimization policy comprises at least one of the following optimization policies: enabling a Nagle algorithm on a TCP connection corresponding to the application, preferentially using the processing resource by a comparatively important thread in the process corresponding to the cold start, reading a file page required for cold starting up the application in advance, performing class verification in advance, and decoding a picture in advance, and loading a basic database in advance. 10 15 20 25 30

17. A device for cold starting up an application, wherein the device includes a processing circuit, a communications interface, and a storage medium, the storage medium stores an instruction, the communications interface is configured to perform information exchange with another device based on the instruction delivered by the processor, and the processing circuit is configured to run the instruction stored in the storage medium, to implement the method according to any one of claims 1 to 8. 35 40

18. The device according to claim 17, wherein the device is a terminal. 45

19. A computer readable storage medium, comprising an instruction, wherein when the instruction is run on a computer, the computer is enabled to perform the method according to any one of claims 1 to 8. 50

55

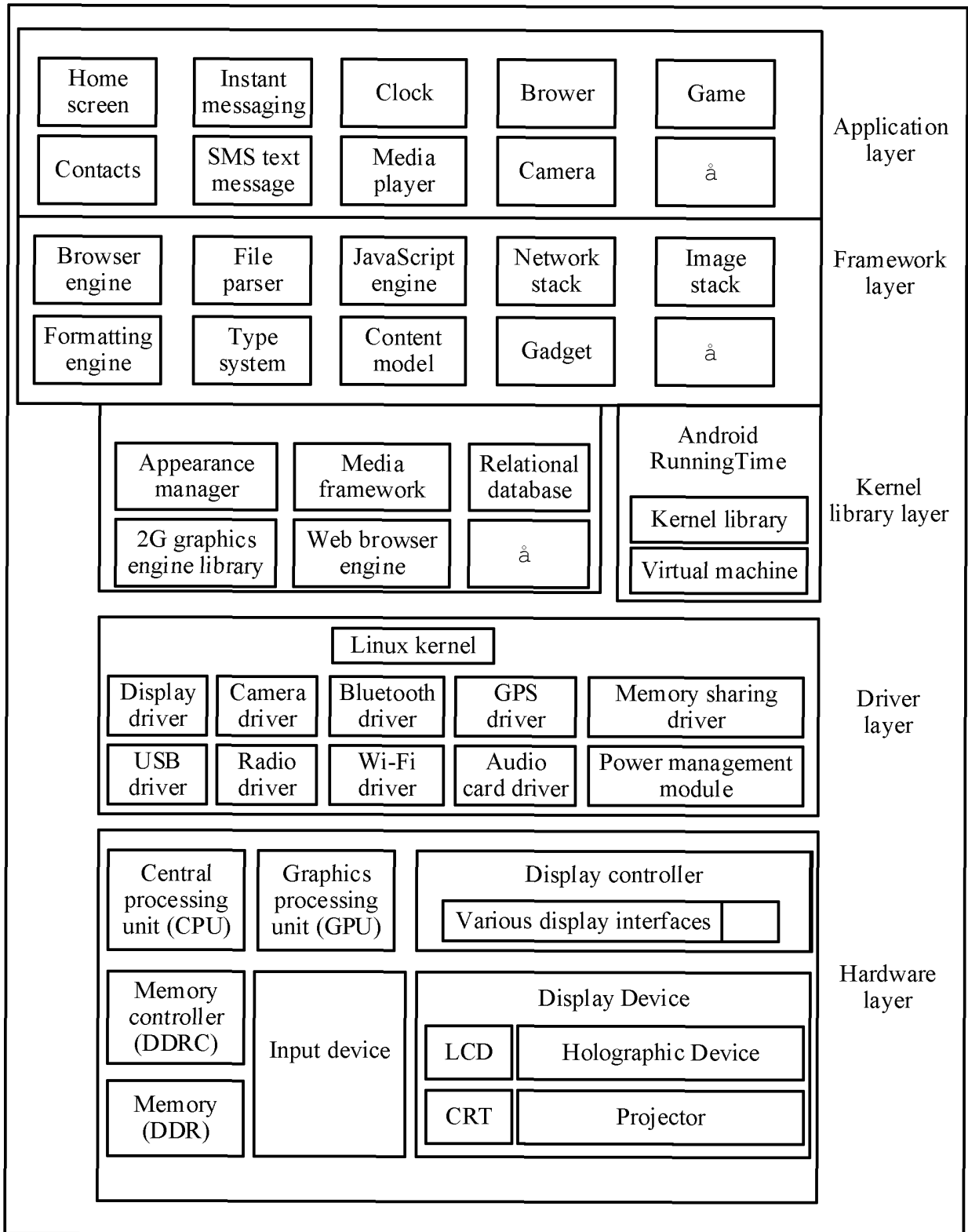


FIG. 1

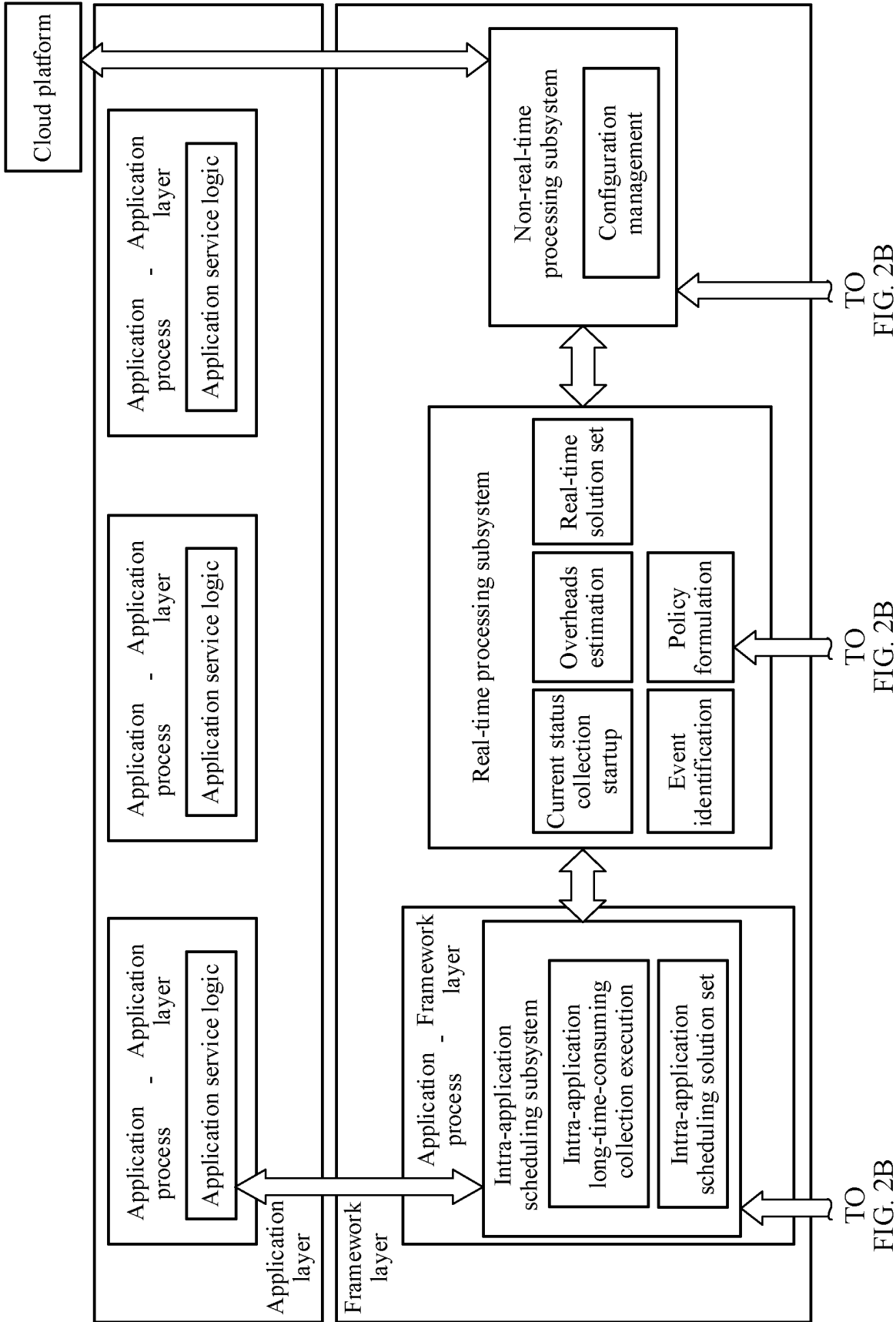


FIG. 2A

CONT. FROM
FIG. 2A

CONT. FROM
FIG. 2A

CONT. FROM
FIG. 2A

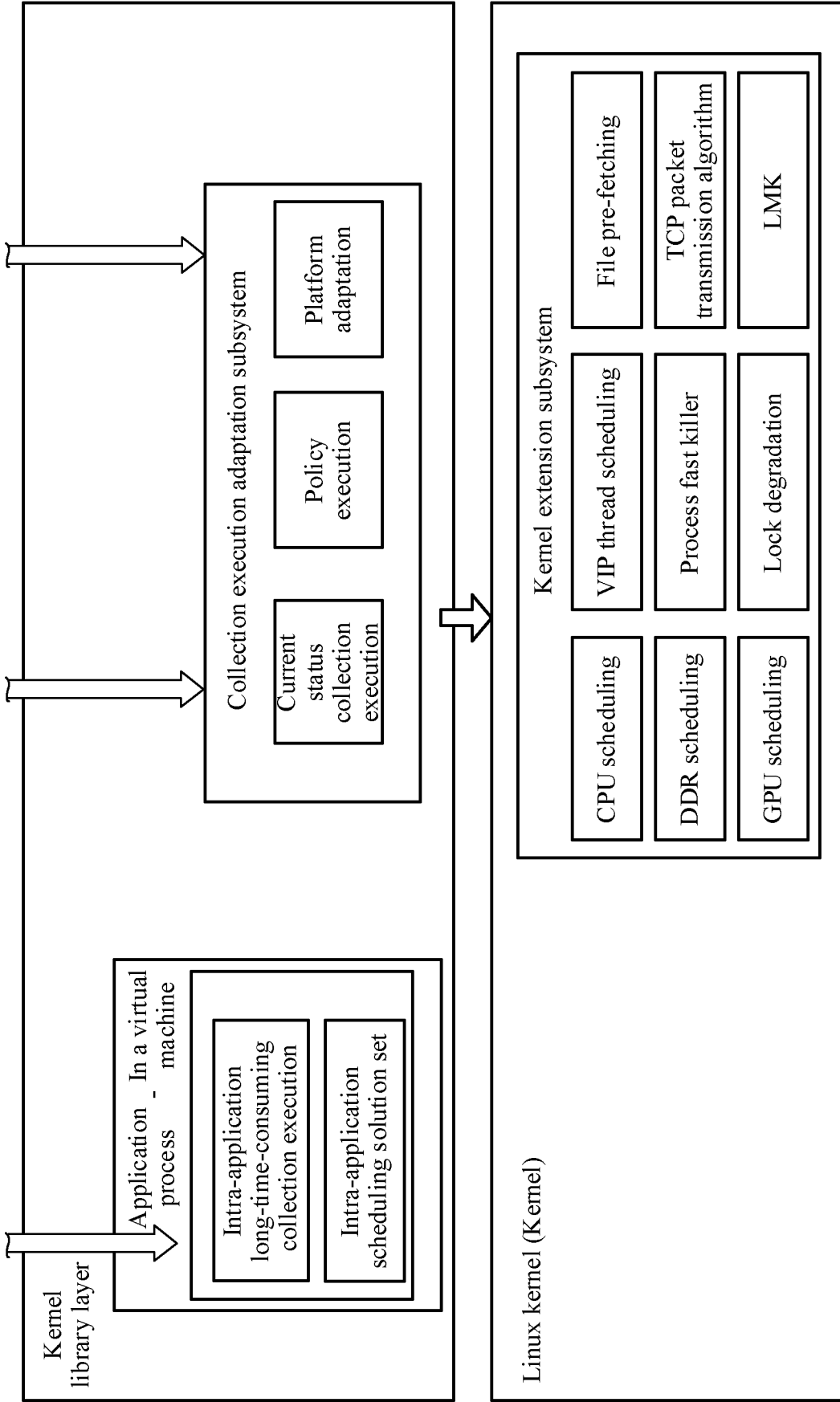


FIG. 2B

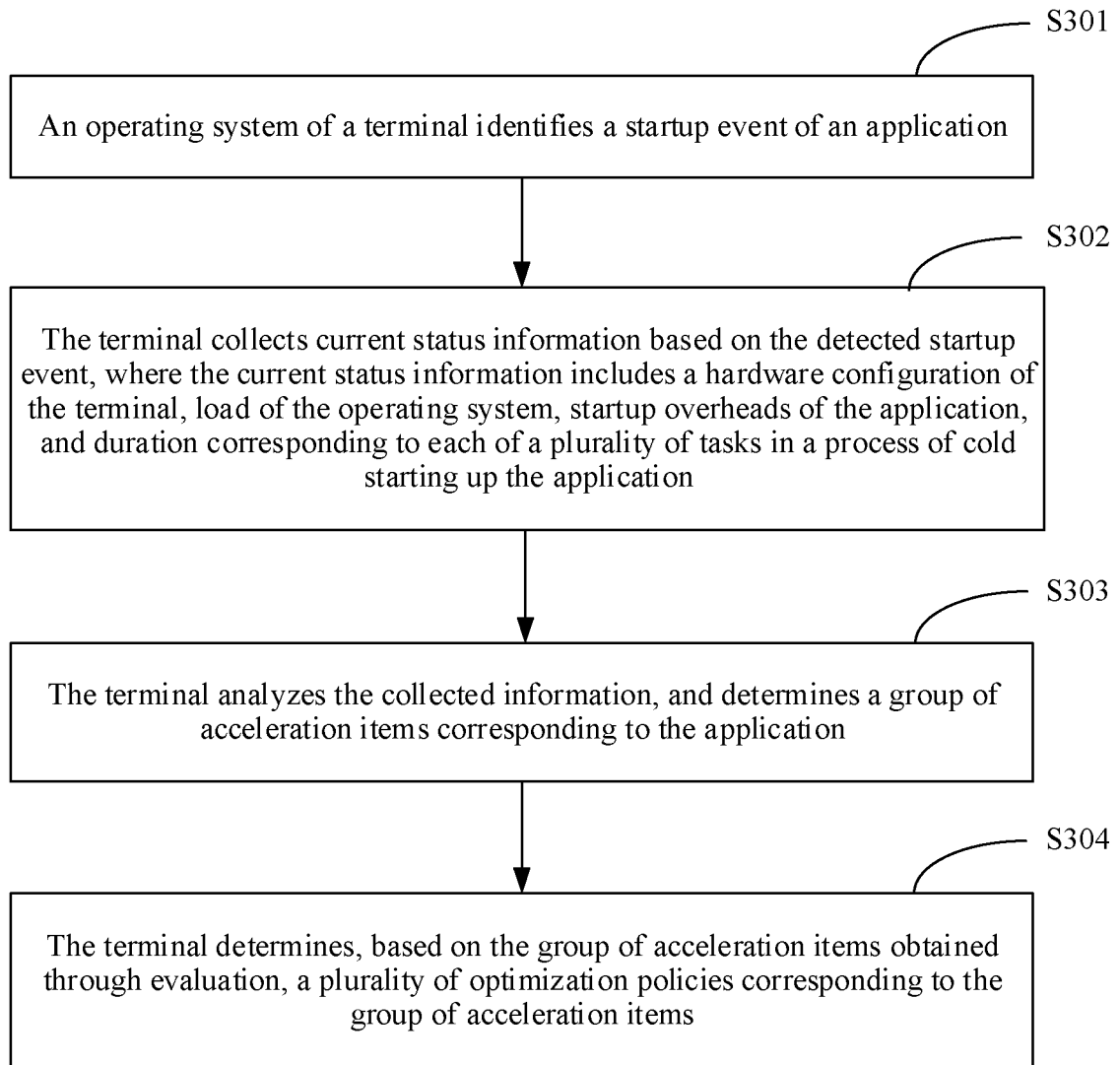


FIG. 3

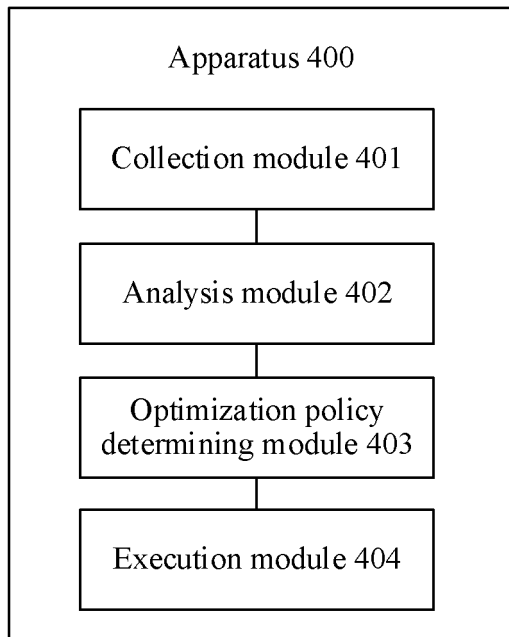


FIG. 4

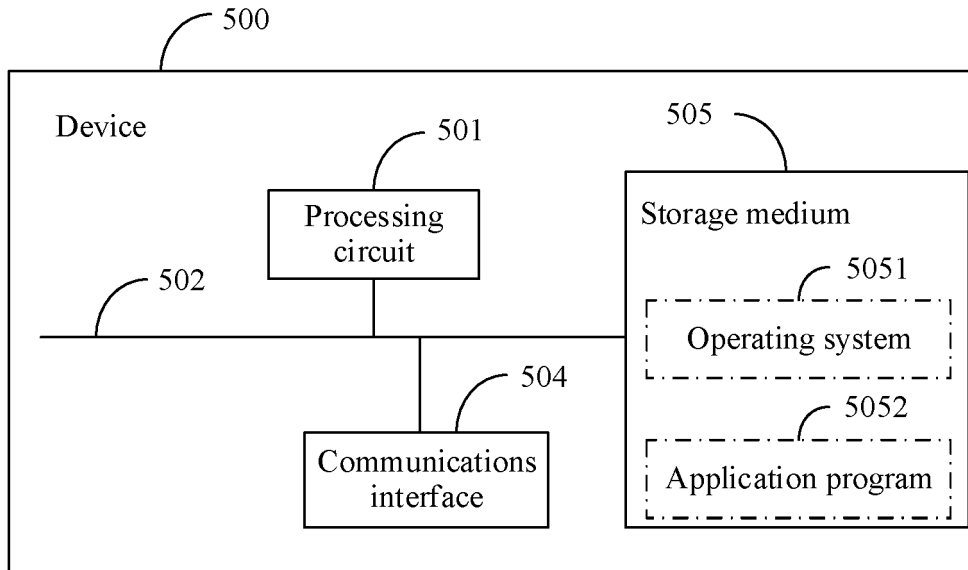


FIG. 5

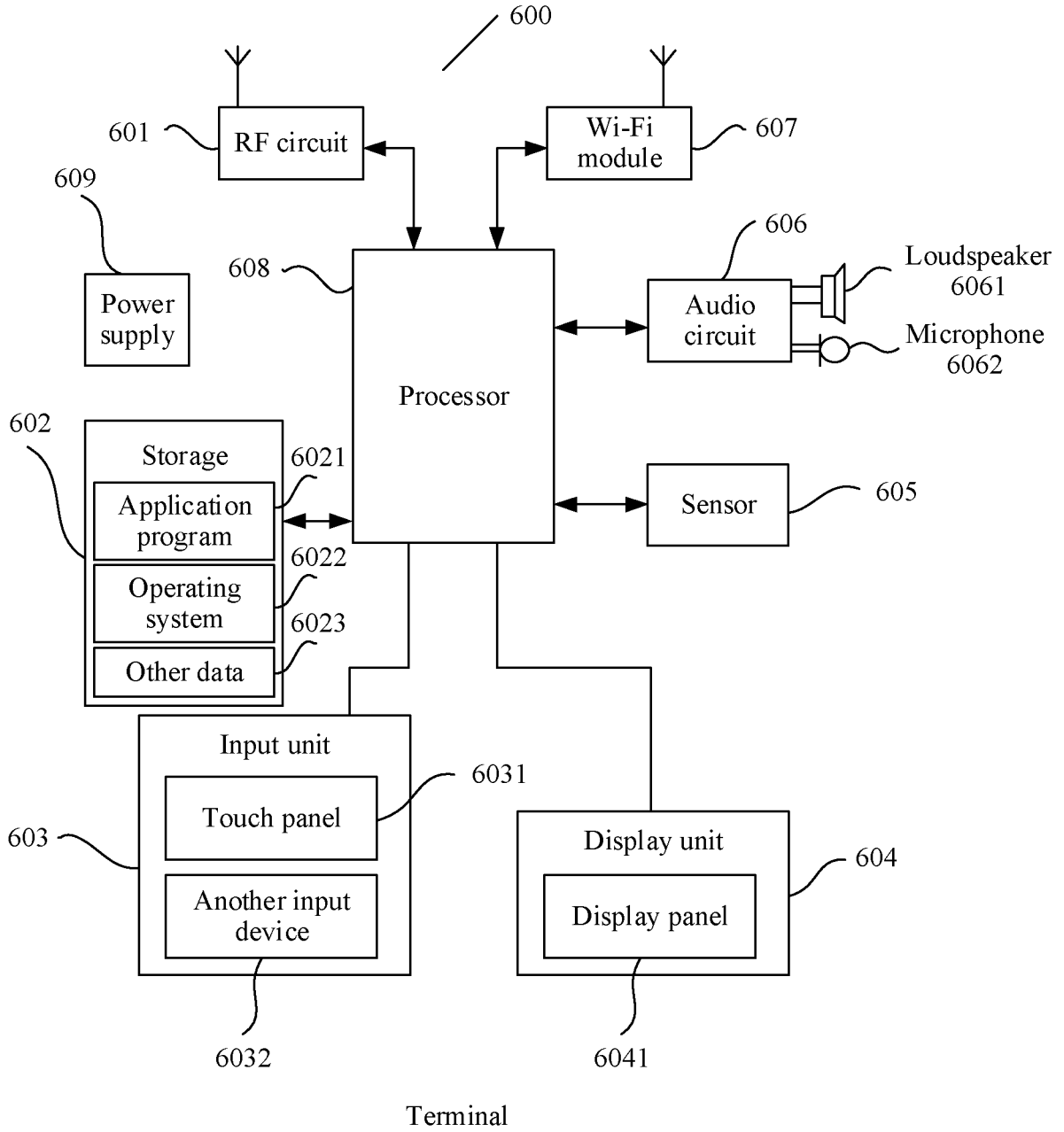


FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2019/111015

5

A. CLASSIFICATION OF SUBJECT MATTER		
G06F 9/445(2018.01)i		
According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED		
Minimum documentation searched (classification system followed by classification symbols)		
G06F		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched		
Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)		
CNPAT; WPI; EPODOC; BAIDU; CNKI; ISI: 应用, 程序, 冷启动, 加速, 提速, 快速, 速度, 优化, 策略, 配置; application, app, cold start, cold boot, speed up, optimize, strategy, configuration		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 106681772 A (SHENZHEN TINNO WIRELESS TECHNOLOGY CO., LTD.) 17 May 2017 (2017-05-17) description, paragraphs [0036]-[0041]	1-19
A	CN 108762833 A (BEIJING ANYUN CENTURY TECHNOLOGY CO., LTD.) 06 November 2018 (2018-11-06) entire document	1-19
A	CN 105630543 A (BEIJING QIHOO TECHNOLOGY CO., LTD., et al.) 01 June 2016 (2016-06-01) entire document	1-19
A	CN 106445617 A (WUHAN DOUYU NETWORK TECHNOLOGY CO., LTD.) 22 February 2017 (2017-02-22) entire document	1-19
A	WO 2013163903 A1 (HUIZHOU TCL MOBILE COMMUNICATION CO., LTD.) 07 November 2013 (2013-11-07) entire document	1-19
<input checked="" type="checkbox"/> Further documents are listed in the continuation of Box C. <input checked="" type="checkbox"/> See patent family annex.		
* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "&" document member of the same patent family	
"A" document defining the general state of the art which is not considered to be of particular relevance		
"E" earlier application or patent but published on or after the international filing date		
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)		
"O" document referring to an oral disclosure, use, exhibition or other means		
"P" document published prior to the international filing date but later than the priority date claimed		
Date of the actual completion of the international search	Date of mailing of the international search report	
03 December 2019	31 December 2019	
Name and mailing address of the ISA/CN	Authorized officer	
China National Intellectual Property Administration (ISA/CN) No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088 China		
Facsimile No. (86-10)62019451	Telephone No.	

Form PCT/ISA/210 (second sheet) (January 2015)

50

55

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2019/111015

5

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 106547598 A (SICHUAN CHANGHONG ELECTRIC CO., LTD.) 29 March 2017 (2017-03-29) entire document	1-19

10

15

20

25

30

35

40

45

50

55

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2019/111015

5
10
15
20
25
30
35
40
45
50
55

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	106681772	A	17 May 2017	None			
CN	108762833	A	06 November 2018	None			
CN	105630543	A	01 June 2016	CN	105630543	B	05 March 2019
CN	106445617	A	22 February 2017	None			
WO	2013163903	A1	07 November 2013	CN	102707973	B	09 December 2015
				CN	102707973	A	03 October 2012
CN	106547598	A	29 March 2017	None			